SMARTTEC PROTOCOL

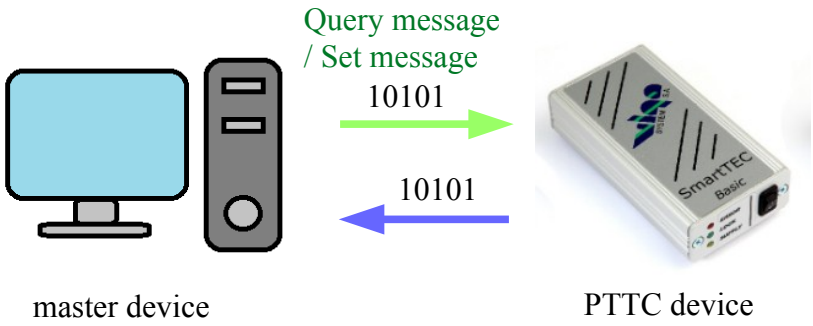
Types of messages

SMARTTEC protocol defines three kinds of messages:

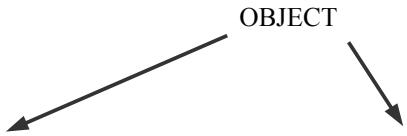
1) “Query message” - data sent from master device to PTTC device - to withdraw a parameter value from PTTC

2) “Set message” - data sent from master device to PTTC device – to set the value of the PTTC parameter

3) “Response message” - data sent from PTTC device to master device – to return the value of the PTTC parameter



Every message is composed of objects. The figure shows the types of object in SMARTTEC protocol.



CONTAINER OBJECT	BASIC OBJECT
Desired to include the set of objects. The objects, however, may belong to the container group, as well as the basic object group.	Desired to store one of data: cstr, int8, uint8, int16, uint16, int32, uint32, float, date_time, serial, bool

The “OBJECT” is made of:

1) OBJ_ID – field of object identifier, each object has own unique number

2) DLEN – field of data length, size of object

3) DATA – field includes one basic data type or container. DATA field includes one of these two components

a) one of the basic data: cstr, int8, uint8, int16, uint16, int32, uint32, float, date_time, serial, bool (Fig. a),

b) encapsulated data - container for other objects. It contains several other objects. (Fig. b).

The following figure shows construction of object.

Fig. a) Object with basic data. This kind of object store basic data type.

BASIC OBJECT - defined for transmitting one of basic data		
OBJ_ID - unique object number	**DLEN** - size of object	**DATA: f.g int8** - one of basic data

Fig. b) Object includes several other objects.

CONTAINER OBJECT - defined to encapsulate data		
OBJ_ID - unique object number	**DLEN** - size of object	**DATA: container** - includes several other objects

***OBJECT**		
OBJ_ID	**DLEN**	**DATA**
***OBJECT**		
OBJ_ID	**DLEN**	**DATA**
(...)		

*Order of objects in container is not predetermined. Identification of objects is based on OBJ_ID value.

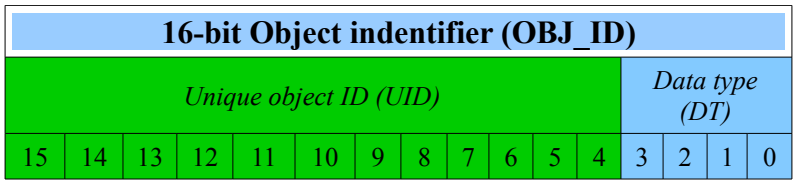
16-bit OBJ_ID - Object identifier format:

Object identifier (OBJ_ID) is composed of two parts: *Unique object ID* and D*ate type*. The object identifier includes two information:

4 bits - *Date type*

12 bits - *Unique object ID*

The least significant four bits are the *Date type*. The next twelve bits indicate the *Unique object ID*.



*Unique object ID (*UID) - each object determines its own unique number. UID number allows to recognize the object.

Data type (DT): List of data types is described below:

Basic types of DATA:

No	data type	Storage size of data type	Size of whole object	Discription
0	container			object encapsulate other objects
1	cstr			size of the field determines the length of the string (variable field size)
2	int8	1 byte	5 bytes	signed char
3	uint8	1 bytes	5 bytes	unsigned char
4	int16	2 bytes	6 bytes	signed short integer
5	uint16	2 bytes	6 bytes	unsigned short integer
6	int32	4 bytes	8 bytes	signed long integer
7	uint32	4 bytes	8 bytes	unsigned long integer
8	float	4 bytes	8 bytes	floating-point, standard, 4 bytes

9	date_time	8 bytes	12 bytes	8 bytes:
				`struct`
				`{`
				`unsigned short int ms;`
				`unsigned char sec;`
				`unsigned char min;`
				`unsigned char hour;`
				`unsigned char day;`
				`unsigned char mon;`
				`unsigned char year; //current year - 1900`
				`}`
				
				Example. Data received from object *MODULE_IDEN_PROD_DATE*: 0xFFFF|FF|FF|FF|01|08|74

MODULE_IDEN_PROD_DATE						
milliseconds (2 bytes)	second (1 byte)	minutes (1 byte)	hour (1 byte)	day (1 byte)	Month (1 byte)	production year = value + 1900 (1 byte)
FFFF	FF	FF	FF	01	08	74
				1	VIII	2016 = 116 + 1900

<table>
<tr><td></td><td></td><td></td><td></td><td>Result = 01-08-2016</td></tr>
<tr><td>10</td><td>serial</td><td>4 bytes</td><td>8</td><td>SMARTEC protocol defines two methods for encoding the serial code. The coding methods depend on value of the first bit. In the first method, the first bit is 1. In the second method, the first bit is 0.
In the first method the serial code includes a serial number (six digits) and a year of its production (two digits). In the second method, the serial code include only a serial number which has nine digits. The serial code in the second method does not include information about the year of production.

Method description

Method 1. Serial code includes serial number and year of manufacture. The first 8 bits describe information about the year of production. The last 24 bits present serial number. The first bit is 0.

Bit sequence: serial code: YYXXXXXX
year: YY
serial number: XXXXXX

bit no. = 1.............................32
100F423F = 00010000000011110100001000111111
year = 00010000 //dec.16
serial number = 000011110100001000111111 //dec.999999

Method 2. Serial code includes only serial number. The AND multiplier is used to set the first bit and to store variable value in the device memory. The AND multiplier has value 7FFFFFF. The AND multiplier could extend the serial number to the length of 9 digits. The first bit is 1.

Calculations:
bit no. = 1.............................32
BB9AC9FF = 10111011100110101100100111111111 //stored value
AND
7FFFFFFF = 01111111111111111111111111111111 //multiplier
result: ---------------------------------
3B9AC9FF = 00111011100110101100100111111111 //read value

BB9AC9FF - value stored in memory
3B9AC9FF - value that the user wants to save</td></tr>
<tr><td>11</td><td>bool</td><td>1 byte</td><td>5 bytes</td><td>binary Value / takes 1 byte</td></tr>
</table>

Size of object

Information about the length of the object is stored in DLEN. The length of the basic object is determined by sum of OBJ_ID, DLEN and DATA. OBJ_ID and DLEN are constant, each of them always have 2 bytes length. Only DATA size can have different length (the length is determined by data type). The size of the object is expressed by the following formula:

size of object = (OBJ_ID + DLEN + size of DATA)
= (2 bytes + 2 bytes + size of DATA)

The possible lengths of data types are described in the fallowing table:

No	data type	size of data type	Size of object
0	container	depends of the contained object	
1	cstr	string length	
2	int8	1 byte	5 bytes
3	uint8	1 bytes	5 bytes
4	int16	2 bytes	6 bytes
5	uint16	2 bytes	6 bytes
6	int32	4 bytes	8 bytes
7	uint32	4 bytes	8 bytes
8	float	4 bytes	8 bytes
9	date_time	8 bytes	12 bytes
10	serial	4 bytes	8 bytes
11	bool	1 byte	5 bytes

Size of basic object is:

The fallowing table illustrate the length of the particular parts of basic object.

BASIC OBJECT (OBJ_ID + DLEN size of DATA) = 5 bytes		
OBJ_ID	**DLEN**	**DATA: int8**
xFFFF	xFFFF	xFF
2 bytes	2 bytes	1 bytes

DATA - int8 value is stored in 1 byte, so that DATA takes 1 byte too.

The length of basic object is 5 bytes, because it is the sum of: OBI_ID = 2 bytes, DLEN = 2 bytes and DATA =1 bytes

Size of object container:

Information about the length of the object is stored in DLEN. The length of the object container is determined by sum of OBJ_ID, DLEN and DATA. OBJ_ID and DLEN are constant, each of them always have 2 bytes length. Only DATA size can have different length (the length is determined by size of basic object).

Size of object container is:

The fallowing table illustrate the length of the particular parts of basic object:

<table>
<tr><td colspan="5">OBJECT CONTAINER
(OBJ_ID + DLEN + size of DATA = 12 bytes)</td></tr>
<tr><td rowspan="2">OBJ_ID</td><td rowspan="2">DLEN</td><td colspan="3">DATA</td></tr>
<tr><td colspan="3">BASIC OBJECT
(OBJ_ID + DLEN + DATA = 8 bytes)</td></tr>
<tr><td></td><td></td><td>OBJ_ID</td><td>DLEN</td><td>DATA:</td></tr>
<tr><td>xFFFF</td><td>xFFFF</td><td>xFFFF</td><td>xFFFF</td><td>(int32 value)
xFFFF
FFFF</td></tr>
<tr><td>2 bytes</td><td>2 bytes</td><td>2 bytes</td><td>2 bytes</td><td>4 bytes</td></tr>
</table>

Length of basic object is 8 bytes, because it is the sum of:

OBJ_ID = 2 bytes, DLEN = 2 bytes and DATA = 4 bytes.

Length of object container is 12 bytes, because it is the sum of:

OBJ_ID = 2 bytes, DLEN = 2 bytes and DATA = BASIC OBJECT = 8 bytes.

Command

Command is the container consisting of the prefix: SET or GET. Command can only be transmitted by the master device – usually the PC. If it is operating properly, PTTC device will immediately respond to the command. PTTC device doesn't provide the queued response, therefore, before sending the new command, the master device (PC) should wait until the response is sent. The safe timeout to wait for the answer is around 500 ms – longer delay indicates an error during the communication.

Query commands are special case of objects. These commands are containers without encapsulated data. For this reason field "DATA" is empty. Construction of commands types is described below:

QUERY COMMAND (4 bytes)	
OBJ_ID	DLEN

SET COMMAND (4 bytes + size of DATA)		
OBJ_ID	DLEN	DATA

RESPONSE COMMAND (4 bytes + size of DATA)		
OBJ_ID	DLEN	DATA

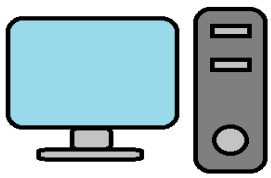
As written above, commands are assigned to the master device. The PTTC response is always a pure container.

Example of set command and response command is:

MASTER device sends the SET_SERVICE_MODE command, which includes object : SERVICE_MODE. SERVICE_MODE object encapsulates another object: SERVICE_MODE_ENABLE (true/false).

PTTC response consists of an object: SERVICE_MODE, encapsulating the SERVICE_MODE_ENABLE object (with the structure as described below).

<table>
<tr><td colspan="6">set command:</td></tr>
<tr><td colspan="6">SET_SERVICE_MODE</td></tr>
<tr><td rowspan="5">OBJ_ID</td><td rowspan="5">DLEN</td><td colspan="4">DATA</td></tr>
<tr><td colspan="4">SERVICE_MODE</td></tr>
<tr><td rowspan="4">OBJ_ID</td><td rowspan="4">DLEN</td><td colspan="3">DATA</td></tr>
<tr><td colspan="3">SERVICE_MODE_ENABLE</td></tr>
<tr><td rowspan="2">OBJ_ID</td><td rowspan="2">DLEN</td><td>DATA</td></tr>
<tr><td>True/False</td></tr>
</table>



master device

PTTC device

<table>
<tr><td colspan="6">response command</td></tr>
<tr><td colspan="6">SERVICE_MODE</td></tr>
<tr><td rowspan="4">OBJ_ID</td><td rowspan="4">DLEN</td><td colspan="4">DATA</td></tr>
<tr><td colspan="4">SERVICE_MODE_ENABLE</td></tr>
<tr><td rowspan="2">OBJ_ID</td><td rowspan="2">DLEN</td><td>DATA</td></tr>
<tr><td>True/False</td></tr>
</table>

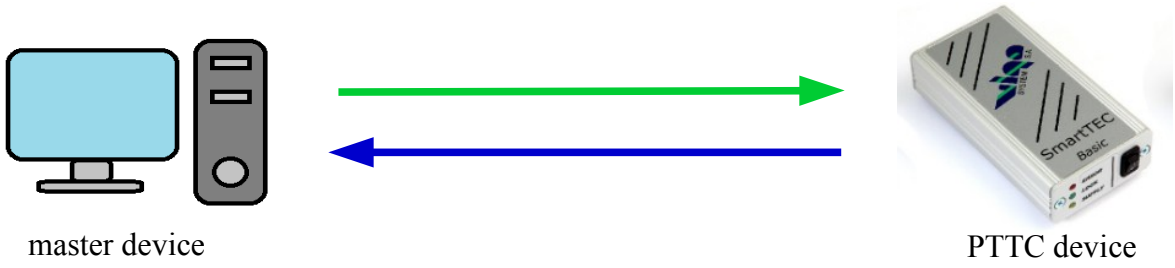
Set messages includes commands with “SET” prefix.

Example of query command and response command is:

MASTER device sends the GET_SERVICE_MODE command.

PTTC response consists of an object: SERVICE_MODE, encapsulating the SERVICE_MODE_ENABLE object (with the structure as described above).

get command:	
GET_SERVICE_MODE	
OBJ_ID	DLEN



master device PTTC device

response command				
SERVICE_MODE				
OBJ_ID	DLEN	DATA		
		SERVICE_MODE_ENABLE		
		OBJ_ID	DLEN	DATA
				True/False

Query messages includes commands with “GET” prefix.

Header file “SmartTec_def.h”

All objects of the SMARTTEC protocol are defined in the header file “SmartTec_def.h”. In this file user can find all commands. Values of OBJ_IDs are calculated on *Unique object ID* and *Data type*. OBJ_ID values are in decimal notation. An illustrative part of header file is:

```
//-------------- Automatically generated file - DO NOT EDIT !
//-------------------------------------------------------------
//-- Creation date: 2016-03-21 10:27:32 ---------------------
//-------------------------------------------------------------
#ifndef SMARTTEC_H
#define SMARTTEC_H
//-------------------------------------------------------------
#define GET_SERVICE_MODE 1024
#define SET_SERVICE_MODE 1040
(...)
#define SERVICE_MODE 4096
#define SERVICE_MODE_ENABLE 4123
(...)
```

Commands are objects containers with prefix GET/SET.

GET_SERVICE_MODE – command is part of query message, OBJ_ID = 1024 UID=64 DT=0 decimal notation
SET_SERVICE_MODE – command is part of set message, OBJ_ID = 1040 UID=65 DT=0 decimal notation
SERVICE_MODE – object container for service_mode_enable, OBJ_ID = 4096 UID=256 DT=0 decimal notation
SERVICE_MODE_ENABLE - basic object OBJ_ID = 4123 UID=257 DT=3 decimal notation
DT=3 means it is basic object which contains unsigned char (uint8) variable.

Remember to make conversion of OBJ_ID from decimal to hexadecimal numbers. All objects are in hexadecimal notation in protocol frame. For ease of use all commands included in header file are shown in the tables “commands list”.

Description of protocol frame

Protocol frame is composed of four elements:
1) START BYTE – start byte marks the beginning of frame
2) DATA FIELD - data field includes one or more objects
3) CRC FIELD – data verification field. The CRC code is calculated on “data field”. CRC code is used to detect accidental changes in transmitted data.
4) STOP BYTE – stop byte marks the end of frame

Fields describe transmitted data in sequential order: the first is START BYTE, the seconds is DATA FIELD, the next is CRC FIELD and the last is STOP BYTE. Each byte in DATA FIELD and CRC FIELD includes

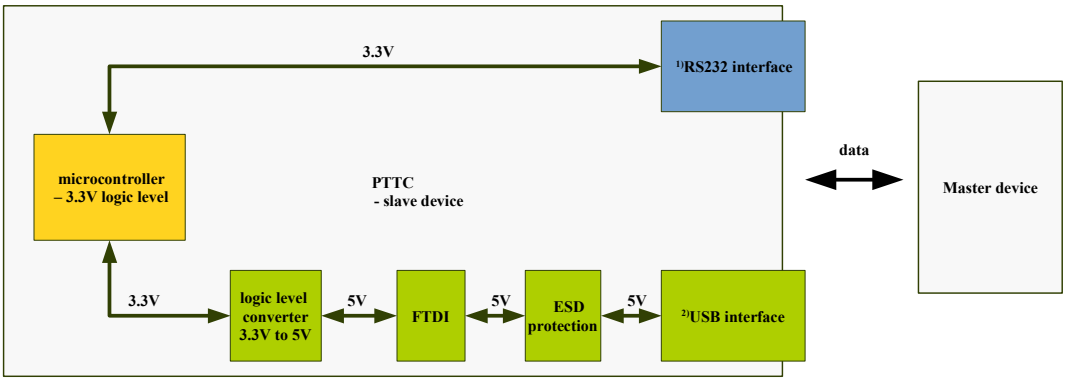
alphanumeric characters. The characters are hexadecimal representation of byte values. The DATA FIELD may include one or several objects.

The following table shows an example of a message frame. The "container" object includes several objects. The container object describes in example two objects named “variable X” and “variable Y”. Order of objects in container is not determined because each object has a unique number of OBJ_ID. OBJ_ID is used to identify objects instead of order. The CRC code is calculated on “data field”. This code is designed to protect data field against accidental changes during transmission.

<table>
<tr><th colspan="12">MESSAGE – example of protocol frame</th></tr>
<tr><td>START BYTE</td><td colspan="9">DATA FIELD</td><td>CRC FIELD</td><td>STOP BYTE</td></tr>
<tr><td rowspan="4">$</td><td colspan="9">OBJECT ‘container’ (4 bytes + size of DATA ‘container’)</td><td rowspan="4">CRC is calculated from DATA FIELD</td><td rowspan="4">#</td></tr>
<tr><td rowspan="3">OBJ_ID</td><td rowspan="3">DLEN</td><td colspan="7">DATA ‘container’</td></tr>
<tr><td colspan="3">OBJECT 'variable X'</td><td rowspan="2">(...)</td><td colspan="3">OBJECT 'variable N'</td></tr>
<tr><td>OBJ_ID</td><td>DLEN</td><td>DATA</td><td>OBJ_ID</td><td>DLEN</td><td>DATA</td></tr>
<tr><td>0x24</td><td colspan="2">0xVALUE</td><td colspan="3">0xVALUE</td><td>(...)</td><td colspan="3">0xVALUE</td><td>0xCRC</td><td>0x23</td></tr>
<tr><td>1 byte</td><td>2 bytes</td><td>2 bytes</td><td>2 bytes</td><td>2 bytes</td><td>size of DATA</td><td>(...)</td><td>2 bytes</td><td>2 bytes</td><td>size of DATA</td><td>2 bytes</td><td>1 byte</td></tr>
</table>

Transmission settings

All versions of PTTC device have USB interface. OEM version of PTTC device also has RS-232 interface. For the implementation of the USB interface FTDI chip is used. If the master device is PC, it is necessary to install Virtual Seria Port COM Driver (VSP COM Driver). VSP COM Driver works and behaves exactly like RS-232 interface - emulating all their settings. Only version of PTTC-OEM board contains pinheads with RS-232 interface outputs (RXD,TXD). RS-232 interface uses 3.3V logic levels. The following figure shows the block diagram of interfaces: USB and RS-232.



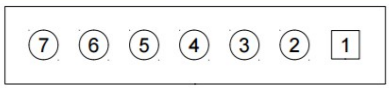
[*1)] Only OEM version of PTTC device

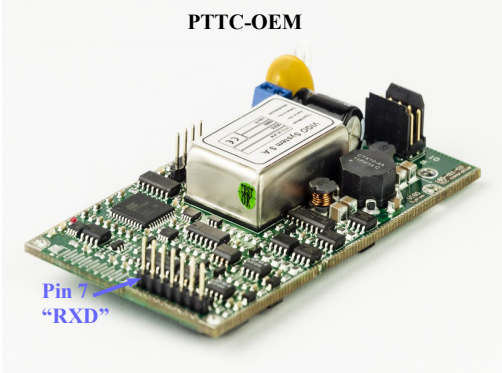
[*2)] All versions of PTTC device

The following figure describes RS-232 outputs pins:

Status / DATA Connector (PTCC-01-OEM)

Pin Number	Symbol	Function
1	ERR – LED	error indicator
2	LOCK – LED	temperature control loop lock indicator
3	SUP – LED	module power supply on indicator
4	3.3 V	auxiliary supply
5	TXD	transmitted data (RS-232)
6	GND	common (signal) ground (RS-232)
7	RXD	received data (RS-232)





Communications settings are:

baud rate: 57600
data bits: 8
stop bit: 1
Parity: none
Flow control: none

Virtual Serial Port COM Driver works exactly like real RS232 interface. It emulates all real serial ports settings and provides strict baud-rate emulation.

Device LED status

Five LEDs indicators are located on the PTTC-OEM version board. LEDs are located on the top of the PCB board. LEDs indicators are used to display the diagnostic status of TEC controller, the outputs of power supply lines and the power source connectivity. The following table shows the functions of LEDs.

LED	Colour	Symbol	Function
solid	blue	power	PTTC is connected to the power source
solid	green	lock	detector is cooled to the required temperature - temperature control loop lock indicator
solid	yellow	sup	negative and positive power supply lines are active (DUBOX2x5 connector, pin 9 „V+", pin 6 „V-")
solid	red	err	detector is not cooled - error indicator
blinking	orange	not programmed	LED is blinking during the normal work mode

CRC-16

CRC is a method of error detecting during the data transmission. In SMARTTEC protocol checksum is written in two bytes. Checksum is calculated only on the basic of the DATA FIELD.
Checksum Crc-16 is known also as: CRC-16-ANSI, CRC-16-IBM etc. The CRC polynomial is used to define the checksum. CRC polynomial: 0x8005. CRC includes 16 bits and it is calculated on the basic of the function presented in "CRC addition" (last pages of manual).

Analysis of part of header file "SmartTec_def.h"

Part of file 'SmartTec_def.h' is:

```
/*-----------  QESTION -----------*/
#define GET_SMARTTEC_CONFIG 1280
(…)
/*-----------  RESPONS -----------*/
#define SMARTTEC_CONFIG  6144
#define SMARTTEC_CONFIG_VARIANT 6163
#define SMARTTEC_CONFIG_VARIANT_MIN 0
#define SMARTTEC_CONFIG_VARIANT_MAX 2
#define SMARTTEC_CONFIG_VARIANT_VALUES "Basic#OEM#Advanced"
#define SMARTTEC_CONFIG_NO_MEM_COMPATIBLE 6187
```

Shown above, part of header file "SmartTec_def.h" includes four objects with similar part name "SMARTTEC_CONFIG". Based on OBJ_ID user can recognize that two objects are containers (1280, 6144) and two objects are store basic data (6163, 6187). Object "GET_SMARTTEC_CONFIG" is container (Data type=0) and has prefix "GET", so it is command. "GET" prefix means that command is part of the query message. Second container object SMARTTEC_CONFIG is part of response message because it is without prefix.

Response object container SMARTTEC_CONFIG includes two objects with basic data:
1) SMARTTEC_CONFIG_VARIANT, in this object is stored uint8 variable "VARIANT". "VARIANT" has range value from 0 to 2. "VARIANT" specifies one of the version of device; 0=BASIC, 1=OEM, 2=Advanced.
2) SMARTTEC_CONFIG_NO_MEM_COMPATIBLE, in this object is stored bool variable "SMARTTEC_CONFIG _NO_MEM_COMPATIBLE".

Based on the analyzed code user can write the query message and the read response message. For ease of use all commands and response included in the header file are shown in the tables "commands list" and "object list".

Example of part of table "Command list" is:

Command „GET/SET"	OBJ_ID value	OBJ_ID		Argument	Device response	Description
		UID	DT			
GET_SMARTTEC_CONFIG	1280	80	0	- - - -	SMARTTEC_CONFIG	(...)

Example of part of table “Order list” is:

Type of object		OBJ_ID value	OBJ_ID			Range of data	Coma position / SI unit	Description
Name of container object	Name of basic data object		UID	DT	Name of data type			
SMARTTEC_CONFIG		6144	384	0	container	n/a	(...)	(...)
	SMARTTEC_CONFIG_VARIANT	6163	385	3	uint8	0 to 2	(...)	(...)
	SMARTTEC_CONFIG _NO_MEM_COMPATIBLE	6187	386	11	bool	true=1 false=0	(...)	(...)

TRANSMISSION CODE FOR RS-232

File “rs232.c”

SET

Message received from master device is: “$0510|0012|1800000501|182B000500|DD84

Parent: 0510 (SET_SERVICE_MODE), first child: 1800 (SERVICE_MODE),

GET

Message received from master device is: “$0500|0004|0F01#”

Parent: 0500 (GET_SERVICE_MODE), first child: (SERVICE_MODE)

```
char *rs232_data[DATA_MAX_LEN];                          //buffer for received message
rs232_get_from_port(rs232_data);(data_rec_pc.data)       //function dependent on the
rs232_send_from_port(rs232_data);                        //implementation of the rs232
                                                         //service

SM_GetId(rs232_data);                                    //retrieves the object id from the
                                                         argument, 2 bytes

SMARTOBJECT* SM_child = SM_GetFirstChild(rs_232_data);   //exposes child object from parent
                                                         container object
uint8_t rs232_rec_action()
{
(...)
   switch (SM_GetId(rs232_data))                         //retrieves parent ID,
                                                         //in this case the parent is a command
   {
   case SET_SMARTTEC_CONFIG:                             //case statement checks id of parent
       if ((SM_GetId(SM_child) == SMARTTEC_CONFIG)       //function SM_GetId retrieves child id,
                                                         //statement if confirms id of the child,
       {
              SmarttecConfig_Load(&dev_smarttec_config, SM_child); //function exposes two basic
                                                                   //objects from a child object
              FLAG_CMD_SET(FLAG_EEPR_WR_SMARTTEC_CONFIG);          //sets flag and writes two
variables the                                                                  //EEPROM:
                                                                   //SMARTTEC_CONFIG_VARIANT and
                                                                 //SMARTTEC_CONFIG_NO_MEM_COMPATIBLE
              FLAG_CMD_SET(FLAG_EEPR_RD_SMARTTEC_CONFIG);          //sets a flag and sends response
for                                                                //command
```

```
        }
    case GET_SMARTTEC_CONFIG:                                   //retrieves parent ID
        FLAG_CMD_SET(FLAG_RS232_SMARTTEC_CONFIG);           //sets a flag and sends response for
command
        break;
    default:
    break;
    }                                                           //end of switch
(...)

    data_rec_pc.status = RS232_BUF_EMPTY;
    return 1;
}

uint8_t rs232_send_action()
{
        int buf_size = 0;

        if (RS232_CHECK_IF_SENDING(data_send_pc) || !FLAG_CMD_CHECK(FLAG_RS232_ALL))
        return 0;                                                   //RS232 busy check, function
                                                                    //dependent of the implementation
                                                                    //of rs232 service
        if (FLAG_CMD_CHECK(FLAG_RS232_DEVICE_IDEN))                 //response for another command
        {
                (...)
        }
        else if (FLAG_CMD_CHECK(FLAG_RS232_SERVICE_MODE))           //answer function for
                                                                    //SERVICE_MODE command
        {
        device_iden    buf_size = ServiceMode_Make(&dev_service_mode, (uint8_t *) data_send_pc.data,
                       RS232_BUF_MAX_LEN_NO_CRC);                           //
        }

        FLAG_CMD_CLR(FLAG_RS232_ALL);                               //flag reset

        if (buf_size > 0)
        {
                frame_make_small_dev((struct buf_rec *)&data_send_pc, buf_size); //function adds CRC
                rs232_pc_packet_send();                             //function dependent of the
                                                                    //implementation of rs232 service
        }

        return 1;

}
```

uint8_t rs232_send_action()

```
typedef struct _SMARTTECCONFIG
{
  uint8_t Variant;
  bool NoMemCompatible;
} SMARTTECCONFIG, *LPSMARTTECCONFIG;

void SmarttecConfig_Clear(SMARTTECCONFIG *ptr);
bool SmarttecConfig_Load(SMARTTECCONFIG *ptr, SMARTOBJECT obj);
SMARTSIZE SmarttecConfig_GetSize(SMARTTECCONFIG *ptr);
SMARTSIZE SmarttecConfig_Make(SMARTTECCONFIG *ptr, uint8_t *buf, SMARTSIZE bufsize);
```

Example of query and response message

Example of query message is:

Query message is sent by user to PTTC device. The device answer is response message.

Query message: ***$050000040F01#***

QUERY MESSAGE				
START	**FIELD DATA**		**CRC**	**STOP**
$	OBJECT GET_SMARTTEC_CONFIG		16 bit crc is calculated from field data	#
	OBJ_ID	DLEN		
24	0500	0004	0F01	23
1 byte	2 bytes	2 bytes	2 bytes	1 byte

hex code: 24|0500|0004|0F01|23
dec code: 36|1280|0004|3841|35

start bit: 24 '$'
field data: 0500|0004, OBJ_ID|DLEN
16 bit crc: 0F01
stop bit: 23 '#'

FIELD DATA includes only two boxes: OBJ_ID and DLEN. In this case, the query message does not include DATA box. The CRC code is calculated on the basis of “FIELD DATA”.

Example of response message to query message is:

SMARTTEC_CONFIG 6144 – command is sent by PTTC to user. It is a main part of response frame.

Response message:

$1800000E
1813000501
182B000500
D80B#

RESPONSE MESSAGE										
START	FIELD DATA								CRC	STOP
$	OBJECT SMARTTEC_CONFIG								16 bit crc	#
	OBJ_ID	DLEN	DATA							
			OBJECT SMARTTEC_CONFIG _Variant			OBJECT SMARTTEC_CONFIG _NoMemCompatible				
			OBJ_ID	DLEN	DATA	OBJ_ID	DLEN	DATA		
24	1800	000E	1813	0005	01	182B	0005	00	D80B	23
1 byte	2 bytes	2 bytes	2 bytes	2 bytes	1 byte	2 bytes	2 bytes	1 byte	2 bytes	1 byte

hex code: 24|1800|000E|1813|0005|01|182B|0005|00|D80B|23
dec code: 36|6144|0014|6163|0005|01|6187|0005|00|55307|35

The response frame includes three objects:
- SMARTTEC_CONFIG (container)
- SMARTTEC_CONFIG_Variant
- SMARTTEC_CONFIG_NoMeMCompatible.

Each object is composed of three boxes: OBJ_ID, DLEN, DATA.

The SMARTTEC_CONFIG object is a container of two objects: SMARTTEC_CONFIG_Variant and SMARTTEC_CONFIG_NoMemCompatible. These two objects are located in DATA box of SMARTEC_CONFIG object .

The CRC code is calculated on the basis of "FIELD DATA".

OBJ_ID. OBJ_ID is based on *Data type* and *Unique object ID.* The following table shows method for calculating

Object name	*OBJ_ID*			*OBJ_ID =* *(12bits) UID|Data type (4bits)*
	UID *Unique object ID*	*DT* *Data typ*		
SMARTTEC_CONFIG	000110000000 dec 384	container	0000 dec 0	0x1800 = 000110000000|0000 dec 6144
SMARTTEC_CONFIG _Variant	000110000001 dec 385	uint8	0011 dec 3	0x1813 = 000110000001|0011 dec 6163
SMARTTEC_CONFIG _NoMemCompatible	000110000010 dec 386	bool	1011 dec 11	0x128B =000110000010|1011 6187

Example of set and response message

Example of set message is:

SET_SMARTTEC_CONFIG 1296 - command is used to set configuration of SMARTEC type (Basic/OEM/Adavanced) and memory compatiblety. It is a main part of set frame.

Set message : $05100012
1800000E
1813000501
182B000500
DD84#

SET MESSAGE							
START	**FIELD DATA**					**CRC**	**STOP**
$	OBJECT 'SET_SMARTTEC_CONFIG' (18 bytes)					16 bit crc	#
	OBJ_ID	DLEN	DATA				
			OBJECT 'SMARTEC _CONFIG' (14 bytes)				
			OBJ_ID	DLEN	DATA		
					objects with basic data		
24	0510	0012	1800	000E	1813000501182B000500	DD84	25
1 byte	2 bytes	2 bytes	2 bytes	2 bytes	10 bytes	2 bytes	1 byte

hex code: 24|0510|0012|*1800|000E|1813|0005|01|182B|0005|00|DD84|25*
dec code: 36|1296|0018|6144|0014|6163|0005|01|6187|0005|00|56708|37

OBJECT SMARTTEC_ CONFIG_VARIANT (5 bytes)			OBJECT SMARTTEC_ CONFIG_NO_MEM _COMPATIBLE (5 bytes)		
OBJ_ID	DLEN	DATA	OBJ_ID	DLEN	DATA
1813	0005	01	182B	0005	00
2 bytes	2 bytes	1 byte	2 bytes	2 bytes	1 byte

Example of response message to query message:

Response message
(SMARTTEC_CONFIG): *$1800000E*
1813000501
182B000500
D80B#

RESPONSE MESSAGE									
START	**FIELD DATA**							**CRC**	**STOP**
$	OBJECT SMARTTEC_CONFIG							16 bit crc	#
	OBJ_ID	DLEN	DATA						
			OBJECT SMARTTEC_CONFIG _Variant			OBJECT SMARTTEC_CONFIG _NoMemCompatible			
			OBJ_ID	DLEN	DATA	OBJ_ID	DLEN	DATA	
24	1800	000E	1813	0005	01	182B	0005	00	D80B	23
1 byte	2 bytes	2 bytes	2 bytes	2 bytes	1 byte	2 bytes	2 bytes	1 byte	2 bytes	1 byte

hex code: 24|*1800|000E|1813|0005|01|182B|0005|00|D80B|25*
dec code: 36|6144|0014|6163|0005|01|6187|0005|00|55307|37

SMARTTEC_CONFIG_VARIANT - range 0...2. 0 = Basic, 1 = OEM , 2 = Advanced. Variable is used to determine the version of SMARTTEC controller.
SMARTTEC_CONFIG_NO_MEM_COMPATIBLE - true or false value.

Types of memory to store data

SmartTEC controller communications with IR module could be in three configurations:

1) standard IR module withour memory EEPROM. Basic device settings are stored in SmartTEC memory.
2) standard IR module with built-in 1-wire memmory. Basic device settings are stored in 1-wire memmory.
3) module SMIPDC-F-200 - communication via RS232 line, half-duplex. Data stored in 1-wire memmory.

Commands list

Query messages includes commands with “GET” prefix.

Set messages includes commands with “SET” prefix and requires arguments.

Response messages includes object container from “device response” field.

Common commands – for all types of modules

Command „GET/SET”	OBJ_ID value	OBJ_ID		Argument	Device response	Description
		UID	DT			
GET_SERVICE_MODE	1024	64	0	- - - - - -	SERVICE_MODE	command is used to check if service mode is enabled /disabled.
SET_SERVICE_MODE	1040	65	0	SERVICE_MODE	SERVICE_MODE	command is used to set service mode, enabled/disable
SET_TRANSPARENT_MODE	1104	69	0	TRANSPARENT_MODE	TRANSPARENT_MODE	command is used to set transparent mode, enabled/disable
GET_DEVICE_IDEN	32	10	0	- - - - - -	DEVICE_IDEN	command is used to read configuration data
SET_DEVICE_IDEN	48	11	0	DEVICE_IDEN	DEVICE_IDEN	command is used to set and save configuration data

Commands for module without memory - configuration is stored in the PTTC controller.

Command „GET/SET”	OBJ_ID value	OBJ_ID		Argument	Device response	Description
		UID	DT			
GET_SMARTTEC_CONFIG	1280	80	0	- - - - - -	SMARTTEC_CONFIG	command is used to read controller configurations data
SET_SMARTTEC_CONFIG	1296	81	0	SMARTTEC_CONFIG	SMARTTEC_CONFIG	command is used to set and save controller configurations data
GET_SMARTTEC_MONITOR	1312	82	0	- - - - - -	SMARTTEC_MONITOR	command is used to read controller data configurations
GET_SMARTTEC_MOD_NO_MEM_IDEN	1536	96	0	- - - - - -	MODULE_IDEN	command is used to read configuration data
SET_SMARTTEC_MOD_NO_MEM_IDEN	1552	97	0	MODULE_IDEN	MODULE_IDEN	command is used to set and save configuration data
GET_SMARTTEC_MOD_NO_MEM_DEFAULT	1568	98	0	- - - - - -	MODULE_BASIC_PARAMS	command is used to read default configuration data
SET_SMARTTEC_MOD_NO_MEM_DEFAULT	1584	99	0	MODULE_BASIC_PARAMS	MODULE_BASIC_PARAMS	command is used to set and save default configuration data
GET_SMARTTEC_MOD_NO_MEM_USER_SET	1600	100	0	- - - - - -	MODULE_BASIC_PARAMS	command is used to read user settings
SET_SMARTTEC_MOD_NO_MEM_USER_SET	1616	101	0	MODULE_BASIC_PARAMS	MODULE_BASIC_PARAMS	command is used to set and save user settings
GET_SMARTTEC_MOD_NO_MEM_USER_MIN	1632	102	0	- - - - - -	MODULE_BASIC_PARAMS	command is used to read minimum settings
SET_SMARTTEC_MOD_NO_MEM_USER_MIN	1648	103	0	MODULE_BASIC_PARAMS	MODULE_BASIC_PARAMS	command is used to set and save minimum settings
GET_SMARTTEC_MOD_NO_MEM_USER_MAX	1664	104	0	- - - - - -	MODULE_BASIC_PARAMS	command is used to read maximum settings
SET_SMARTTEC_MOD_NO_MEM_USER_MAX	1680	105	0	MODULE_BASIC_PARAMS	MODULE_BASIC_PARAMS	command is used to set and save maximum settings

Commands for module with memory - configuration is stored in the module memory

Command „GET/SET”	OBJ_ID value	OBJ_ID		Argument	Device response	Description
		UID	DT			
GET_MODULE_IDEN	2048	128	0	- - - - - -	MODULE_IDEN	command is used to read controller configuration data
SET_MODULE_IDEN	2064	129	0	MODULE_IDEN	MODULE_IDEN	command is used to set and save controller configuration
GET_MODULE_DEFAULT	2112	132	0	- - - - - -	MODULE_BASIC_PARAMS	command is used to read default configurations
SET_MODULE_DEFAULT	2128	133	0	MODULE_BASIC_PARAMS	MODULE_BASIC_PARAMS	command is used to set and save default configurations
GET_MODULE_USER_SET	2144	134	0	- - - - - -	MODULE_BASIC_PARAMS	command is used to read basic user settings
SET_MODULE_USER_SET	2160	135	0	MODULE_BASIC_PARAMS	MODULE_BASIC_PARAMS	command is used to set and save basic user settings
GET_MODULE_USER_MIN	2176	136	0	- - - - - -	MODULE_BASIC_PARAMS	command is used to read minimum basic settings
SET_MODULE_USER_MIN	2192	137	0	MODULE_BASIC_PARAMS	MODULE_BASIC_PARAMS	command is used to set and save minimum basic settings
GET_MODULE_USER_MAX	2208	138	0	- - - - - -	MODULE_BASIC_PARAMS	command is used to read minimum basic settings
SET_MODULE_USER_MAX	2224	139	0	MODULE_BASIC_PARAMS	MODULE_BASIC_PARAMS	command is used to set and save minimum basic settings

Commands for module *SMIPDC* - configuration is stored in the module memory

Command „GET/SET”	OBJ_ID value	OBJ_ID		Argument	Device response	Description save in /read from
		UID	DT			
GET_MODULE_SMIPDC_MONITOR	2560	160	0	- - - - - -	MODULE_SMIPDC_MONITOR	command is used to read controller data configuration
GET_MODULE_SMIPDC_DEFAULT	2688	168	0	- - - - - -	MODULE_SMIPDC_PARAMS	command is used to read default configuration
SET_MODULE_SMIPDC_DEFAULT	2704	169	0	MODULE_SMIPDC_PARAMS	MODULE_SMIPDC_PARAMS	command is used to set and save default configuration
GET_MODULE_SMIPDC_USER_SET	2720	170	0	- - - - - -	MODULE_SMIPDC_PARAMS	command is used to read configuration
SET_MODULE_SMIPDC_USER_SET	2736	171	0	MODULE_SMIPDC_PARAMS	MODULE_SMIPDC_PARAMS	command is used to set and save configuration
GET_MODULE_SMIPDC_USER_MIN	2752	172	0	- - - - - -	MODULE_SMIPDC_PARAMS	command is used to read minimum settings
SET_MODULE_SMIPDC_USER_MIN	2768	173	0	MODULE_SMIPDC_PARAMS	MODULE_SMIPDC_PARAMS	command is used to set and save minimum settings
GET_MODULE_SMIPDC_USER_MAX	2784	174	0	- - - - - -	MODULE_SMIPDC_PARAMS	command is used to read maximum settings
SET_MODULE_SMIPDC_USER_MAX	2800	175	0	MODULE_SMIPDC_PARAMS	MODULE_SMIPDC_PARAMS	command is used to set and save maximum settings
LOAD_MODULE_SMIPDC_PARAMS	2880	180	0	MODULE_USER_SET_BANK	MODULE_USER_SET_BANK	command is used to load user configuration from PIP-DC memory. Configuration can be load from one of four memory banks.
STORE_MODULE_SMIPDC_PARAMS	2896	181	0	MODULE_USER_SET_BANK	MODULE_USER_SET_BANK	command is used to save user configuration to PIP-DC memory. Configuration can be save to one of four memory banks.

Object list

Type of object		OBJ_ID value	OBJ_ID			Range of data	Coma position / SI unit	Description
Name of container object	Name of basic data object		UID	DT	Name of data type			
DEVICE_IDEN		256	16	0				includes basic data objects: OBJ_ID 277 and 361,
	DEVICE_IDEN_TYPE	277	17	5	uint16	0...65535		Describes type of module
	DEVICE_IDEN_FIRM_VER	293	18	5	uint16	0...65535		Describes version of firmware
	DEVICE_IDEN_HARD_VER	309	19	5	uint16	0...65535		Describes version of hardware
	DEVICE_IDEN_NAME	321	20	1	cstr	size 32		Describes module name
	DEVICE_IDEN_SERIAL	346	21	10	serial			Describes module serial number
	DEVICE_IDEN_PROD_DATE	361	22	9	date_time			Describes date of prodution
DEVICE_CHECK		512	32	0				includes basic data objects: OBJ_ID 533
DEVICE_CHECK _VALUE		533	33	5	uint16	0...65535		
SERVICE_MODE		4096	256	0	container	n/a		Disables hardware security; e.g. it ignores short-circuit protection in TEC and thermistor and ignores the time limit to cool detector to required temperature. The device in service mode ignores the warning of status code. Include basic data objects: OBJ_ID 4123 (SMARTTEC_MONITOR_U_TEC). In this mode it is possible to overwrite the default configuration.
	SERVICE_MODE_ENABLE	4123	257	3	bool	true=1 false=0		Responsible for operation state of service mode, true = enable service mode, false = disable service mode.
TRANSPARENT_MODE		5120	320	0	container	n/a		includes basic data objects: OBJ_ID 5147 (TRANSPARENT_MODE_ENABLE)
	TRANSPARENT_MODE_ENABLE	5147	321	3	bool	true=1 false=0		Responsible for operation state of transparent mode, true = enable transparent mode, false = disable trans mode.
SMARTTEC_CONFIG		6144	384	0	container	n/a		includes basic data objects: OBJ_ID 6163 and 6187,
	SMARTTEC_CONFIG_VARIANT	6163	385	3	uint8	0 to 2		Determines the version of PTTC device, value 0 means “Basic”, 1 means “OEM”, 2 means “Advanced” version of device.
	SMARTTEC_CONFIG_NO _MEM_COMPATIBLE	6187	386	11	bool	true=1 false=0		Responsible for availability of EEPROM memory, true: device has an EEPROM, false: device does not have an EEPROM
SMARTTEC_MONITOR		7168	448	0	container	n/a		Command is used to measure or read parameters from the memory. include basic data objects, OBJ_ID from 7168 to 7415. The command can only read the values. The values could not be written in this command.

	SMARTTEC_MONITOR_SUP_ON	7195	449	11	bool	true=1 false=0		Checks operation state of power supply lines (Pin 9 and 6 in DUBOX2x5 connector). true = power supply lines are active, false = power supply lines are inactive,
	SMARTTEC_MONITOR_I_SUP_PLUS	7204	450	4	int16	0...20475	2 /mA	Reads current value of positive supply line (Pin 9 DUBOX2x5 connector): variable range 0...20475 corresponds to 0...204.75mA; resolution is 50uA,
	SMARTTEC_MONITOR_I_SUP_MINUS	7220	451	4	int16	-20475 ...0	2 /mA	Reads current value of negative supply line (Pin 6 DUBOX2x5 connector), variable range 0...20475 corresponds to 0...204.75mA, resolution is 50uA,
	SMARTTEC_MONITOR_FAN_ON	7243	452	11	bool	true=1 false=0		Checks operation state of fan output (Pin 7 DUBOX2x5 connector), true = enable fan output, false = disable fan output
	SMARTTEC_MONITOR_I_FAN_PLUS	7252	453	4	int16	0...4095	1 /mA	Reads output current value of fan output (Pin 7 DUBOX2x5 connector), variable range 0...4095 corresponds to 0...409.5 mA, resolution is 100 uA
	SMARTTEC_MONITOR_I_TEC	7268	454	4	int16	0...20475	4 /mA	Reads current value of TEC output, variable range 0...20475 corresponds to 0...2,0475 A, resolution is 0.5 mA
	SMARTTEC_MONITOR_U_TEC	7284	455	4	int16	0...20475	3 /V	Reads output voltage value of TEC, variable range 0...20475 is corresponds to 0...20,475 V, resolution 0.5 mA
	SMARTTEC_MONITOR_U_SUP_PLUS	7300	456	4	int16	0.20475	3 /V	Reads output voltage value of positive supply line, variable range 0...20475 corresponds to 0...20,475 V, resolution 0.5 mV
	SMARTTEC_MONITOR_U_SUP_MINUS	7316	457	4	int16	-20475...0	3 /V	Reads output voltage value of negative supply line, variable range -20475...0 corresponds to 0...20,475 V, resolution 0.5 mV
	SMARTTEC_MONITOR_T_DET	7334	458	6	int32	0...400000	3 /K	Reads detector temperature in Kelvin degree, variable range 0...400000 responds to 0...400 K, 18bits ADC
	SMARTTEC_MONITOR_T_INT	7348	459	4	int16	0.1500	1 /C	Reads detector temperature in Celsius degree, variable range 0...1500 corresponds to 0...150C, 18bits ADC
	SMARTTEC_MONITOR_PWM	7365	460	5	uint16	0...65535		Reads PWM settings of TEC controller, variable range 0...65535. PWM=0 means TEC is not cooling. PWM=65536 means TEC is cooling with maximum power.
	SMARTTEC_MONITOR_STATUS	7379	461	3	uint8			**Status code**: (device.h, soft pttc) 0 – detector is cooled, temperature is equal(-/+ 1 K) to temperature defined by user 1 – during the cooling proces 2 - the cooling is deactivated. Check PTTC settings. **Error code**:

								128 - “detector overheat” - the set temperature could not be reached during 120 second. 129 - Measured current value is higher then maximum current value. PTTC power is off. 130 - TEC circuit open connection 131 - TEC circuit is closed connection 132 - thermistor circuit open connection 133 - thermistor circuit closed connection 134 - the temperature inside PTCC is higher than limit 135 - the connected module without memory is not compatible or no module is connected 136 - memory was detected but there are some communication problem. 137 – PIP data fault, there are some communication problem. 138 - 1-wire data fault, there are some communication problem. 139 - PTTC memory fault 140 - PIP is incompatible 141 - 1-wire memory is incompatible When the error status code appears the re-turn of the PTTC devices might be required.
	SMARTTEC_MONITOR_MODULE_TYPE	7395	462	3	uint8	0...3		Reads type of module: 0 = “NONE” - means module not connected, 1 = “NOMEM” - means module with no configuration memory, 2 = “1WIRE” - means module with internal configuration memory, 3 = “SIMPDC” - means programmable intelligent IR module
	MONITOR_TH_ADC	7415	463	7	uint32		/mV	Reads voltage value of thermistor, 0...2,046V, resolution is 1mV
MODULE_IDEN		8192	512	0	container	n/a		includes basic data objects, OBJ_ID from 8211 to 8581
	MODULE_IDEN_TYPE	8211	513	3	uint8			Describes type of memory: 0 = “NONE”, 1 = “NOMEM”, 2 = “1WIRE”, 3 = “SIMPDC”
	MODULE_IDEN_FIRM_VER	8229	514	5	uint16			Describes version of firmware
	MODULE_IDEN_HARD_VER	8245	515	5	uint16			Describes version of hardware
	MODULE_IDEN_NAME	8257	516	1	cstr	size 32		Describes module name
	MODULE_IDEN_SERIAL	8282	517	10	serial			Describes module serial number
	MODULE_IDEN_DET_NAME	8289	518	1	cstr	size 32		Describes detector name
	MODULE_IDEN_DET_SERIAL	8314	519	10	serial			Describes detector serial number

<table>
<tr><td></td><td>MODULE_IDEN_PROD_DATE</td><td>8329</td><td>520</td><td>9</td><td>date_time</td><td></td><td></td><td>Describes date of manufacture of the module, an example is:
<table>
<tr><th>milli -seconds (2 bytes)</th><th>Second 2 bytes</th><th>Min 2byt.</th><th>Hour 2byt.</th><th>Day 2byt.</th><th>Month 2bytes</th><th>production year = value + 1900 (1 byte)</th></tr>
<tr><td>FFFF</td><td>FF</td><td>FF</td><td>FF</td><td>01</td><td>08</td><td>74</td></tr>
<tr><td></td><td></td><td></td><td></td><td>1</td><td>VIII</td><td>2016 = 116 + 1900</td></tr>
</table></td></tr>
<tr><td></td><td>MODULE_IDEN_TEC_TYPE</td><td>8339</td><td>521</td><td>3</td><td>uint8</td><td></td><td></td><td>Variable range 0-3: 0=NONE, 1=NOMEM, 2=1WIRE, 3=SIMPDC</td></tr>
<tr><td></td><td>MODULE_IDEN_TH_TYPE</td><td>8355</td><td>522</td><td>3</td><td>uint8</td><td></td><td></td><td>Describes thermistor type</td></tr>
<tr><td></td><td>MODULE_IDEN_TEC_PARAM1</td><td>8376</td><td>523</td><td>8</td><td>float</td><td></td><td></td><td>Describes TEC parameters</td></tr>
<tr><td></td><td>MODULE_IDEN_TEC_PARAM2</td><td>8392</td><td>524</td><td>8</td><td>float</td><td></td><td></td><td>Describes TEC parameters</td></tr>
<tr><td></td><td>MODULE_IDEN_TEC_PARAM3</td><td>8408</td><td>525</td><td>8</td><td>float</td><td></td><td></td><td>Describes TEC parameters</td></tr>
<tr><td></td><td>MODULE_IDEN_TEC_PARAM4</td><td>8424</td><td>526</td><td>8</td><td>float</td><td></td><td></td><td>Describes TEC parameters</td></tr>
<tr><td></td><td>MODULE_IDEN_TH_PARAM1</td><td>8440</td><td>527</td><td>8</td><td>float</td><td></td><td></td><td>Describes thermistor parameters</td></tr>
<tr><td></td><td>MODULE_IDEN_TH_PARAM2</td><td>8456</td><td>528</td><td>8</td><td>float</td><td></td><td></td><td>Describes thermistor parameters</td></tr>
<tr><td></td><td>MODULE_IDEN_TH_PARAM3</td><td>8472</td><td>529</td><td>8</td><td>float</td><td></td><td></td><td>Describes thermistor parameters</td></tr>
<tr><td></td><td>MODULE_IDEN_TH_PARAM4</td><td>8488</td><td>530</td><td>8</td><td>float</td><td></td><td></td><td>Describes thermistor parameters</td></tr>
<tr><td></td><td>MODULE_IDEN_COOL_TIME</td><td>8581</td><td>536</td><td>5</td><td>uint16</td><td></td><td></td><td>Responsible for setting maximum time of cooling module. If the module does not reach desired temperature it will be turned off.</td></tr>
<tr><td>MODULE_CHECK</td><td></td><td>8704</td><td>544</td><td>0</td><td>container</td><td>n/a</td><td></td><td>includes basic data objects: OBJ_ID 8725 (MODULE_CHECK_VALUE)</td></tr>
<tr><td></td><td>MODULE_CHECK_VALUE</td><td>8725</td><td>545</td><td>5</td><td>uint16</td><td></td><td></td><td>Command is used to check memory from banks.</td></tr>
<tr><td>MODULE_BASIC_PARAMS</td><td></td><td>9216</td><td>576</td><td>0</td><td>container</td><td>n/a</td><td></td><td>includes basic data objects, OBJ_ID from 9235 to9351</td></tr>
<tr><td></td><td>MODULE_BASIC_PARAMS_SUP_CTRL</td><td>9235</td><td>577</td><td>3</td><td>uint8</td><td>0...2</td><td></td><td>Describes operating modes of power supply lines. Variable range 0...2: 0=AUTO, 1=OFF, 2=ON. In “AUTO” mode configuration, power supply for preamplifier will be enable only when detector is cooled to desired temperature (GREEN LED ligths). OFF/ON disable/enable power supply outputs (state of TEC cooling does not matter).</td></tr>
<tr><td></td><td>MODULE_BASIC_PARAMS_U_SUP_PLUS</td><td>9252</td><td>578</td><td>4</td><td>int16</td><td>3000 ...15000</td><td></td><td>Responsible for setting output voltage value of positive power line (pin9 in DUBOX2x5 connector). Variable range 3000...15000 is corresponds to 3...15V.</td></tr>
<tr><td></td><td>MODULE_BASIC_PARAMS_U_SUP_MINUS</td><td>9268</td><td>579</td><td>4</td><td>int16</td><td>-15000 ...-3000</td><td></td><td>Responsible for setting output voltage value of negative power line (pin 6 in DUBOX2x5 connector). Variable range -15000...-3000 corresponds to -15...-3V.</td></tr>
<tr><td></td><td>MODULE_BASIC_PARAMS_FAN_CTRL</td><td>9283</td><td>580</td><td>3</td><td>uint8</td><td>0...2</td><td></td><td>Describes operation state of fan control (Pin 7 DUBOX2x5 connector). Variable range 0...2. 0=AUTO, 1=OFF, 2=ON. Mode AUTO/ON means fan enable, OFF means fan outputs is disable.</td></tr>
</table>

	MODULE_BASIC_PARAMS_TEC_CTRL	9299	581	3	uint8	0..2		Describes variable range 0...2. 0=AUTO, 1=OFF, 2=ON. Variable is used to set operating modes of TEC cooler.
	MODULE_BASIC_PARAMS_PWM	9317	582	5	uint16	0.65535		Describes PWM settings of TEC. Variable range 0...65535. PWM=0 means TEC is not cooling. PWM=65536 means TEC is cooling with maximum power.
	MODULE_BASIC_PARAMS_I_TEC_MAX	9332	583	8	float	0...20475		Describes maximum current for TEC output. Variable range 0...20475 corresponds to 0...2.0475A.
	MODULE_BASIC_PARAMS_T_DET	9351	584	7	uint32	100000 ...400000		Describes detector temperature in K degree. Variable range 0...400000 responds to 0...400 K, 18bits ADC.
MODULE_USER _SET_BANK		10240	640	0	container	n/a		includes basic data objects: OBJ_ID 10259 (MODULE_USER_SET_BANK_INDEX)
	MODULE_USER_SET_BANK_INDEX	10259	641	3	uint8	0...3		
MODULE_SMIPDC _MONITOR		11264	704	0	container	n/a		includes basic data objects, OBJ_ID from 11284 to 11444
	MODULE_SMIPDC_MONITOR_SUP_PLUS	11284	705	8	float	0.20475		Reads voltage value of positive power line (pin9 in DUBOX2x5 connector). Variable range 3000...15000 is corresponds to 3...15V.
	MODULE_SMIPDC_MONITOR _SUP_MINUS	11300	706	8	float	-20480 ...20470		Reads voltage value of negative power line (pin 6 in DUBOX2x5 connector). Variable range -15000...-3000 corresponds to -15...-3V.
	MODULE_SMIPDC_MONITOR _FAN_PLUS	11316	707	8	float	0...20475		Reads operating mode of fan output (Pin 7 DUBOX2x5 connector). Variable range 0...2. 0=AUTO, 1=OFF, 2=ON. For safety cannot turn off the fan output.
	MODULE_SMIPDC_MONITOR _TEC_PLUS	11332	708	8	float	-20480 ...20470	4 /A	Reads maximum current for TEC positive output. Variable range 0...20475 corresponds to 0...2.0475A.
	MODULE_SMIPDC_MONITOR _TEC_MINUS	11348	709	8	float	-20480 ...20470	4 /A	Reads maximum current for TEC negative output. Variable range 0...20475 corresponds to 0...2.0475A.
	MODULE_SMIPDC_MONITOR_TH1	11364	710	8	float	-2048 ...2047		Reads voltage value of thermistor pin 1.
	MODULE_SMIPDC_MONITOR_TH2	11380	711	8	float	0...2047		Reads voltage value of thermistor pin 2.
	MODULE_SMIPDC_MONITOR_U_DET	11396	712	8	float	-2048 ...2047	3 /V	-2048...2047 corresponds to -2.048...2.047V
	MODULE_SMIPDC_MONITOR_U_1ST	11412	713	8	float	-4096 ...4094	3 /V	-4096...4094 corresponds to -4.096...4.094V
	MODULE_SMIPDC_MONITOR_U_OUT	11428	714	8	float	-10240. ..10235	3 /V	-10.240...10.235 corresponds to -10.240...10.235V
	MODULE_SMIPDC_MONITOR_TEMP	11444	715	8	float	0...1000	1 /C	Reads detector temperature in Celsius degree. Variable range 0...1000 corresponds to 0...400 K, 18bits ADC.
MODULE_SMIPDC _PARAMS		12288	768	0	container	n/a		includes basic data objects, OBJ_ID from 12309 to 12419
	MODULE_SMIPDC_PARAMS_DET_U	12309	769	5	uint16	0...256	/V	Describes value of voltage bias. Variable range 0...256 corresponds to 0-1V.

	MODULE_SMIPDC_PARAMS_DET_I	12325	770	5	uint16	0...256	/mA	Describes value of current bias compensation. Variable range 0...256 corresponds to 0-10mA.
	MODULE_SMIPDC_PARAMS_GAIN	12341	771	5	uint16	0...256		Responsible for setting gain in the second stage. Variable range 0...256..
	MODULE_SMIPDC_PARAMS_OFFSET	12357	772	5	uint16	0...256	/V	Responsible for setting offset value. Variable range 0...256 corresponds to +1...-1V
	MODULE_SMIPDC_PARAMS_VARACTOR	12373	773	5	uint16	0....4095		Responsible for frequency compensation for the preamplifier first stage Variable range 0...4095
	MODULE_SMIPDC_PARAMS_TRANS	12387	774	3	uint8	0...1		Responsible for transimpedance of 1st stage preamplifier. Variable 0 or 1, 0=LOW means 1kOhm, 1=HIGH means 5kOhm
	MODULE_SMIPDC_PARAMS_ACDC	12403	775	3	uint8	0...1		Responsible for the coupling mode. 0 or 1, 0 means AC coupling, 1 means DC coupling
	MODULE_SMIPDC_PARAMS_BW	12419	776	3	uint8	0...2		Describes value of bandwidth. Variable range 0/1/2, 0=LOW means 1.5Mhz, 1=MID means 15Mhz, 2=HIGH means it depends on detector parameters and first stage transimpedance.

GET COMMANDS DESCRIPTIONS FOR MODULE WITHOUT MEMORY

Get command	OBJ_ID	Query message	Response	OBJ_ID	Response message
GET_SERVICE_MODE	1024	*$050000040F01#*	MODULE_SMIPDC_PARAMS		*$10000009* *101B000500* *2E09#*
GET_SMARTTEC_CONFIG	*1280*	*$04000004F300#*	*SMARTTEC_CONFIG*		*$1800000E* *1813000500* *182B000500* *D80B#*
GET_SMARTTEC_MONITOR	1312	*$05200004C500#*	SMARTTEC_MONITOR		*$1C00005E* *1C1B000500* *1C2400060000* *1C3400060000* *1C4B000500* *1C5400060000* *1C6400060000* *1C7400060000* *1C8400060000* *1C9400060000* *1CA6000800000000* *1CB400060000* *1CC500060000* *1CD3000587* *1CE3000500* *1CF7000800010000A* *CEEB#*
GET_SMARTTEC_MOD	1536	*$060000044B01#*	MODULE_IDEN		*$200000C9*

_NO_MEM_IDEN					*2013000500* *202500060000* *203500060001* *204100240000000000000000000000000000000000000000000000000000000000000000* *205A000800000000* *206100240000000000000000000000000000000000000000000000000000000000000000* *207A000800000000* *2089000CFFFFFFFFFF010874* *2093000501* *20A3000501* *20B8000800C0DA44* *20C8000800007041* *20D8000800000000* *20E8000800000000* *20F8000800809243* *2108000800800945* *21180008666E3645* *2128000800000000* *218500060078* *773B#*
GET_SMARTTEC_MOD_NO_MEM_DEFAULT	1568	*$062000048100#*	MODULE_BASIC_PARAMS		*$24000033* *2413000500* *242400062328* *24340006DCD8* *2443000500* *2453000500* *246500060000* *247400061194* *2487000800038270* *7562#* **The remaining three commands have the same data structure.* *For this reason they will not be described in detail.*
GET_SMARTTEC_MOD_NO_MEM_USER_SET	1600	*$064000049F00#*	MODULE_BASIC_PARAMS		
GET_SMARTTEC_MOD_NO_MEM_USER_MIN	1632	*$066000045501#*	MODULE_BASIC_PARAMS		
GET_SMARTTEC_MOD_NO_MEM_USER_MAX	1664	$06800004A300#	MODULE_BASIC_PARAMS		

“GET” COMMANDS DESCRIPTIONS FOR MODULE WITH BUILD-IN MEMORY

Get command	OBJ_ID	Query message	Response	OBJ_ID	Response message
GET_MODULE_IDEN	2048	*$08000004A303#*	MODULE_IDEN		*$200000C9* *2013000502* *202500060001* *203500060001* *204100247A6173696C61637A2032347600000000000000000000000000000000000000000000* *205A000800000000* *20610024000000000000000000000000000000000000000000000000000000000000000000000000* *207A000800000000* *2089000C0000000000100574* *2093000501* *20A3000501* *20B8000800000000* *20C8000800000000* *20D8000800000000* *20E8000800000000* *20F8000800809243* *2108000800800945* *21180008666E3645* *2128000800000000* *218500060078* *BB83#*
GET_MODULE_DEFAULT	*2112*	*$084000047702#*	*MODULE_BASIC_PARAMS*		*$24000033* *2413000500* *242400062EE0* *24340006D120* *2443000501* *2453000500* *246500060000* *247400062EE0* *2487000800038270* *6255#* **The remaining three commands have the same data structure.* *For this reason they will not be described in detail.*
GET_MODULE_USER_SET	2144	***$08600004BD03#***	*MODULE_BASIC_PARAMS*		
GET_MODULE_USER_MIN	2176	***$088000044B02#***	*MODULE_BASIC_PARAMS*		
GET_MODULE_USER_MAX	2208	***$08A000048103#***	*MODULE_BASIC_PARAMS*		

“GET” COMMANDS DESCRIPTIONS FOR MODULE SMIPDC

Get command	OBJ_ID	Query message	Response	Response message
GET_MODULE_SMIPDC_MONITOR	2560	*$0A0000041B02#*	MODULE_SMIPDC _MONITOR	*$2C000046* *2C1400060000* *2C2400060000* *2C3400060000* *2C4400060000* *2C5400060000* *2C6400060000* *2C7400060000* *2C8400060000* *2C9400060000* *2CA400060000* *2CB400060000* *DCBC#*
GET_MODULE_SMIPDC _DEFAULT	*2688*	*$0A800004F303#*	*MODULE_SMIPDC _PARAMS*	*$30000031* *301500060000* *302500060000* *303500060000* *304500060000* *305500060000* *3063000500* *3073000500* *3083000500* *DCE3#*
*GET_MODULE_SMIPDC_USER_SET	2720	$0AA000043902#	MODULE_SMIPDC_PARAMS	
*GET_MODULE_SMIPDC_USER_MIN	2752	$0AC000042702#	MODULE_SMIPDC_PARAMS	
*GET_MODULE_SMIPDC_USER_MAX	2784	$0AE00004ED03#	MODULE_SMIPDC_PARAMS	**The remaining three commands have the same data structure. For this reason they will not be described in detail.*

“SET” COMMANDS DESCRIPTIONS

Get command	OBJ_ID	Query message	Response	Response message
SET_SERVICE_MODE	1040	*$0410000D* *10000009* *101B000501* *6F96#*	SERVICE_MODE	*$10000009* *101B000501* *EEC8#*
SET_TRANSPARENT_MODE	*1104*	*$0450000D* *14000009* *141B000501* *5054#*	*TRANSPARENT_MODE*	*$14000009* *141B000501* *EE0B#*
SET_SMARTTEC_CONFIG	1296	*$05100012* *1800000E* *1813000501* *182B000500* *DD84#*	SMARTTEC_CONFIG	*$1800000E* *1813000501* *182B000500* *D80B#*
SET_SMARTTEC_MOD_NO_MEM_IDEN	1552	$061000CD 200000C9 2013000500 202500060000 203500060001 204100240000000000000000000000 000000000000000000000000000000 00000000000000 204100240000000000000000000000 000000000000000000000000000000 00000000000000 206100240000000000000000000000 000000000000000000000000000000 00000000000000 207A000800000000 2089000CFFFFFFFFFF010874 2093000501 20A3000501 20B8000800C0DA44 20C8000800007041 20D8000800000000 20E800800000000 20F8000800809243 2108000800800945 21180008666E3645		$200000C9 2013000500 202500060000 203500060001 204100240000000000000000000000000000000000000000000000000000000000000000 000000000 205A000800000000 206100240000000000000000000000000000000000000000000000000000000000000000 000000000 207A000800000000 2089000CFFFFFFFFFF010874 209300050120A3000501 20B8000800C0DA44 20C8000800007041 20D8000800000000 20E8000800000000 2108000800800945 21180008666E3645 2128000800000000 218500060078 773B#

		2128000800000000 218500060078 D5C6#		
SET_SMARTTEC_MOD_NO_MEM_DEFAULT	1584	$06300037 24000033 2413000500 242400062328 24340006DCD8 2443000500 2453000500 246500060000 247400061194 2487000800038270 77B0#	MODULE_BASIC_PARAMS	$24000033 2413000500 242400062328 24340006DCD8 2443000500 2453000500 246500060000 247400061194 2487000800038270 7562#
SET_SMARTTEC_MOD_NO_MEM_USER_SET	1616	$06500037 24000033 2413000500 242400062328 24340006DCD8 2443000500 2453000500 246500060000 247400061194 2487000800038270 4A65#		The remaining three commands have the same data structure. For this reason they will not be described in detail.
SET_SMARTTEC_MOD_NO_MEM_USER_MIN	1648	$06700037 24000033 2413000500 242400060BB8 24340006C568 2443000500 2453000500 246500060000 247400060000 248700080002BF20 0AEA#		
SET_SMARTTEC_MOD_NO_MEM_U	1680	$06900037 24000033		

SER_MAX		2413000500 242400063A98 24340006F448 2443000500 2453000500 246500060000 247400062EE0 24870008000493E0 3096#		

16-bits CRC CODE

```
#define CRC16 0x8005

typedef unsigned short int uint16;
typedef unsigned char uint8;

uint16 gen_crc16(const uint8 *data, uint16 size)
{
    uint16 out = 0;
    int bits_read = 0, bit_flag;

    if(data == NULL)
        return 0;

    while(size > 0)
    {
        bit_flag = out >> 15;
        out <<= 1;
        out |= (*data >> (bits_read)) & 1;
        bits_read++;
        if(bits_read > 7)
        {
            bits_read = 0;
            data++;
            size--;
        }
        if(bit_flag)
            out ^= CRC16;
    }

    int i;
    for (i = 0; i < 16; ++i) {
        bit_flag = out >> 15;
        out <<= 1;
        if(bit_flag)
            out ^= CRC16;
    }

    uint16 crc = 0;
    i = 0x8000;
    int j = 0x0001;
    for (; i != 0; i >>=1, j <<= 1) {
        if (i & out) crc |= j;
    }

    return crc;
}
```

GET COMMANDS DESCRIPTIONS IN IR MODULE WITHOUT MEMORY

This chapter describes “GET” commands in IR module without memory. All data are stored in the memory of SMARTTEC controller.

GET_SERVICE_MODE 1024 - command is used to check if service mode is enabled/disabled

Question command: *$04000004F300#*

Response (SERVICE_MODE): *$10000009*
101B000500
2E09#

START $	ALL DATA					CRC-16bit	STOP #
	HEAD 'SERVICE_MODE'		Data 1 'SERVICE_MODE_ENABLE'				
	OBJ_ID	size of all data	OBJ_ID	size of data	value		
24	1000	0009	101B	0005	00	2E09	25

SERVICE_MODE_ENABLE - true/false value. Responsible for enabled/disabled service mode.

GET_SMARTTEC_CONFIG 1280 - command is used to read controller configurations data

Question command: *$050000040F01#*

Response (*SMARTTEC_CONFIG)*: *$1800000E*
1813000501
182B000500
D80B#

START $	ALL DATA			CRC-16bit	STOP #
	HEAD 'SMARTTEC_CONFIG'		DATA 1 - 2		
	OBJ_ID	size of all data			
24	1800	000E		D80B	25

Data 1 (5 bytes) SMARTTEC_ CONFIG_VARIANT			Data 2 (5 bytes) SMARTTEC_ CONFIG_NO_MEM _COMPATIBLE		
OBJ_ID	size of data	value	OBJ_ID	size of data	value
1813	0005	01	182B	0005	00

SMARTTEC_CONFIG_VARIANT - variable is used to determine the version of SMARTTEC controller.

Variable range 0...2. 0 = Basic, 1 = OEM , 2 = Advanced

SMARTTEC_CONFIG_NO_MEM_COMPATIBLE - true or false value.

GET_SMARTTEC_MONITOR 1312 - command is used to read controller data configurations

Question command: *$05200004C500#*

Response (SMARTTEC_MONITOR) : *$1C00005E*
1C1B000500
1C2400060000
1C3400060000
1C4B000500
1C5400060000
1C6400060000
1C7400060000
1C8400060000
1C9400060000
1CA6000800000000
1CB400060000
1CC500060000
1CD3000587
1CE3000500
1CF700080010000A
CEEB#

START $	ALL DATA			CRC-16bit	STOP #
	HEAD 'SMARTTEC_MONITOR'		DATA 1 - 15		
	OBJ_ID	size of all data			
24	1C00	0x005E		CEEB	25

Data (5 bytes) SMARTTEC _MONITOR _SUP_ON			Data (6 bytes) SMARTTEC _MONITOR_ I_SUP_PLUS			Data (6 bytes) SMARTTEC _MONITOR _I_SUP_MINUS			Data (5 bytes) SMARTTEC _MONITOR _FAN_ON			Data (6 bytes) SMARTTEC _MONITOR_ I_FAN_PLUS		
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value
1C1B	0005	00	1C24	0006	0000	1C34	0006	0000	1C4B	0005	00	1C54	0006	00
Data (6 bytes) SMARTTEC _MONITOR			Data (6 bytes): SMARTTEC _MONITOR _U_TEC			Data (6 bytes): SMARTTEC _MONITOR_ U_SUP_PLUS			Data (6 bytes): SMARTTEC _MONITOR_ U_SUP_MINUS			Data (8 bytes): SMARTTEC _MONITOR _T_DET		
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value
1C64	0006	0000	1C74	0006	0000	1C84	0006	0000	1C94	0006	0000	1CA6	0008	0000 0000
Data (6 bytes): SMARTTEC _MONITOR_T_INT			Data (6 bytes): SMARTTEC _MONITOR_PWM			Data (5 bytes): SMARTTEC _MONITOR _STATUS			Data (5 bytes): SMARTTEC _MONITOR _MODULE_TYPE			Data (8 bytes): SMARTTEC _MONITOR _TH_ADC		
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value
1CB4	0006	0000	1CC5	0006	0000	1CD3	0005	87	1CE3	0005	00	1CF7	0008	0010 000A

1) *SMARTTEC_MONITOR_SUP_ON* – true/false value. Responsible for power supply output on/off.

2) *SMARTTEC_MONITOR_I_SUP_PLUS* –used to read current value of output supply plus

variable range 0...20475 responds to 0...204.75mA, resolution 50uA, (Pin 9 DUBOX2x5 connector).

3) *SMARTTEC_MONITOR_I_SUP_MINUS* –used to read current value of output supply minus

variable range -20475...0 responds to 0...204.75mA, resolution 50uA, (Pin 6 DUBOX2x5 connector).

4) *SMARTTEC_MONITOR_FAN_ON* - true/false value. Responsible for EN/DIS fan output (Pin 7 DUBOX2x5 connector).

5) *SMARTTEC_MONITOR_I_FAN_PLUS* - used to read current value of output fan (Pin 7 DUBOX2x5 connector),

variable range 0...4095, responds to 0...409.5 mA, resolution 100 uA

6) *SMARTTEC_MONITOR_I_TEC* – used to read value of TEC current (Thermo Electric Cooling)

variable range 0...20475 responds to 0...2,0475 A, resolution 0.5 mA

7) *SMARTTEC_MONITOR_U_TEC* – used to read voltage of TEC (Thermo Electric Cooling)

variable range 0...20475 responds to 0...20,475 V, resolution 0.5 mV

8) *SMARTTEC_MONITOR_U_SUP_PLUS* - voltage value of output supply plus

variable range 0...20475 responds to 0...20,475 V, resolution 0.5 mV

9) *SMARTTEC_MONITOR_U_SUP_MINUS* - voltage value of output supply plus

variable range -20475...0 responds to -20,475...0 V, resolution 0.5 mV

10) *SMARTTEC_MONITOR_T_DET* – detector temperature in K degree

variable range 0...400000 responds to 0...400 K, 18bits ADC

11) *SMARTTEC_MONITOR_T_INT* – detector temperature in C degree

variable range 0...1500 responds to 0...150C, 18bits ADC

12) *SMARTTEC_MONITOR_PWM* – PWM settings of TEC, variable range 0...65535

13) *SMARTTEC_MONITOR_STATUS* – error code/status code

14) *SMARTTEC_MONITOR_MODULE_TYPE* – variable range 0...3.

0 = “NONE”, 1 = “NOMEM”, 2 = “1WIRE”, 3 = “SIMPDC”

15) *SMARTTEC_MONITOR_TH_ADC* – 0...2,046V, resoltuion 1mV

GET_SMARTTEC_MOD_NO_MEM_IDEN 1536 - command is used to read data in no memory IR module (NOMEM)

Question command: *$060000044B01#*

Response (MODULE_IDEN): *$200000C9*
2013000500
202500060000
203500060001
20410024000000000000000000000000000000000000000000000000000000000000000000000000
205A000800000000
20610024000000000000000000000000000000000000000000000000000000000000000000000000
207A000800000000
2089000CFFFFFFFFFF010874
2093000501
20A3000501
20B8000800C0DA44
20C8000800007041
20D8000800000000
20E8000800000000
20F8000800809243
2108000800800945
21180008666E3645
2128000800000000
218500060078
773B#

START $	ALL DATA			CRC-16bit	STOP #
	HEAD 'MODULE_IDEN'		DATA 1 - 19		
	OBJ_ID	size of all data			
24	2000	00C9		773B	25

Data (5 bytes) MODULE_IDEN _TYPE			Data (6 bytes) MODULE_IDEN _FIRM_VER			Data (6 bytes) MODULE_IDEN _HARD_VER			Data (24 bytes) MODULE_IDEN _NAME			Data (8 bytes) MODULE_IDEN _SERIAL		
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value
2013	0005	00	2025	0006	0000	2035	0006	0001	2041	0024	table	205A	0008	00

Data (24 bytes) MODULE_IDEN _DET_NAME			Data (8 bytes): MODULE_IDEN _DET_SERIAL			Data (12 bytes): MODULE_IDEN _PROD_DATE			Data (5 bytes): MODULE_IDEN _TEC_TYPE			Data (5 bytes): MODULE_IDEN _TH_TYPE		
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value
2061	0024	table	207A	0008	00	2089	000C	table	2093	0005	01	20A3	0005	01

Data (8 bytes) MODULE_IDEN _TEC_PARAM1			Data (8 bytes) MODULE_IDEN _TEC_PARAM2			Data (8 bytes) MODULE_IDEN _TEC_PARAM3			Data (8 bytes) MODULE_IDEN _TEC_PARAM4			Data (8 bytes) MODULE_IDEN _TH_PARAM1		
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value
20B8	0008	00C0 DA44	20C8	0008	0000 7041	20D8	0008	00	20E8	0008	00	20F8	0008	0080 9243

Data (8 bytes) MODULE_IDEN _TH_PARAM2			Data (8 bytes) MODULE_IDEN _TH_PARAM3			Data (8 bytes) MODULE_IDEN _TH_PARAM4			Data (6 bytes) MODULE_IDEN _COOL_TIME		
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value
2108	0008	0080 0945	2118	0008	666E 3645	2128	0008	0000	2185	0006	0078

1) *MODULE_IDEN_TYPE* - variable range 0-3, 0=NONE, 1=NOMEM, 2=1WIRE, 3=SIMPDC
2) *MODULE_IDEN_FIRM_VER* – version of firmware
3) *MODULE_IDEN_HARD_VER* – version of hardware
4) *MODULE_IDEN_NAME* – module name

name	MODULE_IDEN_NAME																							
value	*0x20410024000000000000000000000000000000000000000000000000000000000000000000000000*																							
byte no.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
hex	20	41	0	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
symbol	0x2041		0x0024		NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL
descrip.	OBJ_ID		size		ASCII characters																			

5) *MODULE_IDEN_SERIAL* - module serial number
6) *MODULE_IDEN_DET_NAME* – detector name

name	MODULE_IDEN_DET_NAME																							
value	*0x20610024000000000000000000000000000000000000000000000000000000000000000000000000*																							
byte no.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
hex	20	41	0	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
symbol	0x2061		0x0024		NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL
descrip.	OBJ_ID		size		ASCII characters																			

7) *MODULE_IDEN_DET_SERIAL* – detector serial number
8) *MODULE_IDEN_PROD_DATE* - date of manufacture of the module,|

```
struct
{
        unsigned short int ms;
        unsigned char sec;
        unsigned char min;
        unsigned char hour;
        unsigned char day;
        unsigned char mon;
        unsigned char year;         //current year – 1900, rg. 2016 – 1900 = 116
}
```

milliseconds (2 bytes)	Second (1 byte)	Minutes (1 byte)	Hour (1 byte)	Day (1 byte)	Month (1 byte)	production year = value + 1900 (1 byte)
FFFF	FF	FF	FF	01	08	74
				1	VIII	2016 = 116 + 1900

0xFFFF|FF|FF|FF|01|08|74 = 01-08-2016

9) *MODULE_IDEN_TEC_TYPE* - variable range 0-3, 0=NONE, 1=NOMEM, 2=1WIRE, 3=SIMPDC
10) *MODULE_IDEN_TH_TYPE* – type of thermistor
11) *MODULE_IDEN_TEC_PARAM1* – TEC parameter
12) *MODULE_IDEN_TEC_PARAM2* – TEC parameter
13) *MODULE_IDEN_TEC_PARAM3* – TEC parameter
14) *MODULE_IDEN_TEC_PARAM4* – TEC parameter
15) *MODULE_IDEN_TH_PARAM1* – thermistor parameter
16) *MODULE_IDEN_TH_PARAM2* – thermistor parameter
17) *MODULE_IDEN_TH_PARAM3* - thermistor parameter
18) *MODULE_IDEN_TH_PARAM4*- thermistor parameter
19) *MODULE_IDEN_COOL_TIME* – maximum time to cool a IR module. If the module does not reach desired temperature it will be turned off after 120s.

GET_SMARTTEC_MOD_NO_MEM_DEFAULT 1568 - command is used to read default data in no memory IR module (NOMEM)

Question command: *$062000048100#*

Response (MODULE_BASIC_PARAMS): *$24000033*
2413000500
242400062328
24340006DCD8
2443000500
2453000500
246500060000
247400061194
2487000800038270
7562#

START $	ALL DATA			CRC-16bit	STOP #
	HEAD 'MODULE_BASIC| _PARAMS'		DATA 1 - 8		
	OBJ_ID	size of all data			
24	2400	0033		7562	25

Data (5 bytes) MODULE_BASIC _PARAMS_SUP _CTRL			Data (6 bytes) MODULE_BASIC _PARAMS_U _SUP_PLUS			Data (6 bytes) MODULE_BASIC _PARAMS _U_SUP_MINUS			Data (5 bytes) MODULE_BASIC _PARAMS_FAN _CTRL		
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value
2413	0005	00	2424	0006	2328	2434	0006	DCD8	2443	0005	00
Data (5 bytes) MODULE_BASIC _PARAMS_TEC _CTRL			Data (6 bytes): MODULE_BASIC _PARAMS_PWM			Data (6 bytes): MODULE_BASIC _PARAMS _I_TEC_MAX			Data (8 bytes): MODULE_BASIC _PARAMS_T_DET		
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value
2453	0005	00	2465	0006	0000	2474	0006	1194	2487	0008	0003 8270

1) MODULE_BASIC_PARAMS_SUP_CTRL – variable is used to set operating mode of power supply output. Variable range 0...2. 0=AUTO, 1=OFF, 2=ON. AUTO mode is used to protect the detector. Power supply for preamplifier will be turn on when detector is cooled to desired temperature (GREEN LED indicator). OFF/ON means disable/enable power supply outputs.
2) MODULE_BASIC_PARAMS_U_SUP_PLUS – is used to set plus output voltage for pin 9 in DUBOX2x5 connector. Variable range 3000...15000 corresponds to 3...15V.
3) MODULE_BASIC_PARAMS_U_SUP_MINUS – is used to set minus output voltage for pin 6 in DUBOX2x5 connector. Variable range -15000...-3000 corresponds to -15...-3V.
4) MODULE_BASIC_PARAMS_FAN_CTRL – *f*an control output (Pin 7 DUBOX2x5 connector). Variable range 0...2. 0=AUTO, 1=OFF, 2=ON. Mode AUTO/ON and OFF mean fan enable and disable.
5) *MODULE_BASIC_PARAMS_TEC_CTRL* - Variable range 0...2. 0=AUTO, 1=OFF, 2=ON. Variable is used to set operating mode of TEC cooler.
6) *MODULE_BASIC_PARAMS_PWM* - PWM settings of TEC. Variable range 0...65535.
7) *MODULE_BASIC_PARAMS_I_TEC_MAX* - maximum current for TEC. Variable range 0...20475 corresponds to 0...2.0475A.

8) MODULE_BASIC_PARAMS_T_DET - detector temperature in K degree. Variable range 0...400000 responds to 0...400 K, 18bits ADC.

The remaining three commands have the same data structure. For this reason they will not be described in detail.(*GET_SMARTTEC_MOD_NO_MEM_USER_SET 1600*, *GET_SMARTTEC_MOD_NO_MEM_USER_MIN 1632*, *GET_SMARTTEC_MOD_NO_MEM_USER_MAX 1664*)

GET_SMARTTEC_MOD_NO_MEM_USER_SET 1600 - command is used to read user settings in no memory module (NOMEM)

Question command: *$064000049F00#*

Response (MODULE_BASIC_PARAMS): *$24000033*
2413000500
242400062328
24340006DCD8
2443000500
2453000500
246500060000
247400061194
2487000800038270
7562#

GET_SMARTTEC_MOD_NO_MEM_USER_MIN 1632 - command is used to read minimum settings of no memory module (NOMEM)

Question command: *$066000045501#*

Response (MODULE_BASIC_PARAMS): *$24000033*
2413000500
242400060BB8
24340006C568
2443000500
2453000500
246500060000
247400060000
2487000800O2BF20
215E#

GET_SMARTTEC_MOD_NO_MEM_USER_MAX 1664 – command is used to read maximum settings of no memory module (NOMEM)

Question command: $06800004A300#

Response (MODULE_BASIC_PARAMS): *$24000033*
2413000500
242400063A98
24340006F448
2443000500
2453000500
246500060000
247400062EE0
24870008000493E0
743B#

“GET” COMMANDS DESCRIPTIONS FOR MODULE WITH BUILD-IN MEMORY

This chapter describes “GET” commands in IR module with build in memory. All data are stored in IR module memory (DS24B33). Communications with 1 wire memory are done by pins: DATA 8, GND 3 in DUBOX2x5 connector.

GET_MODULE_IDEN 2048 - command is used to read controller configurations data from module memory

Question command: *$08000004A303#*
Response (MODULE_IDEN):

$200000C9
2013000502
202500060001
203500060001
204100247A6173696C61637A2032347600000000000000000000000000000000000000000000
205A000800000000
20610024000000000000000000000000000000000000000000000000000000000000000000000000
207A000800000000
2089000C0000000000100574
2093000501
20A3000501
20B8000800000000
20C8000800000000
20D8000800000000
20E8000800000000
20F8000800809243
2108000800800945
21180008666E3645
2128000800000000
218500060078
BB83#

START $	ALL DATA			CRC-16bit	STOP #
	HEAD 'MODULE_IDEN'		DATA 1 - 19		
	OBJ_ID	size of all data			
24	2000	00C9		BB83	25

Data (5 bytes) MODULE_IDEN _TYPE			Data (6 bytes) MODULE_IDEN _FIRM_VER			Data (6 bytes) MODULE_IDEN _HARD_VER			Data (5 bytes) MODULE_IDEN _NAME			Data (6 bytes) MODULE_IDEN _SERIAL		
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value
2013	0005	02	2025	0006	0001	2035	0006	0001	2041	0024	table	205A	0008	0000 0000
Data (6 bytes) MODULE_IDEN _DET_NAME			Data (6 bytes): MODULE_IDEN _DET_SERIAL			Data (6 bytes): MODULE_IDEN _PROD_DATE			Data (6 bytes): MODULE_IDEN _TEC_TYPE			Data (8 bytes): MODULE_IDEN _TH_TYPE		
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value
2061	0024	table	207A	0008	0000 0000	2089	000C	*00000000 00100574*	2093	0005	01	20A3	0005	01
Data (6 bytes): MODULE_IDEN _TEC_PARAM1			Data (6 bytes): MODULE_IDEN _TEC_PARAM2			Data (5 bytes): MODULE_IDEN _TEC_PARAM3			Data (5 bytes): MODULE_IDEN _TEC_PARAM4			Data (8 bytes): MODULE_IDEN _TH_PARAM1		
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value
20B8	0008	0000 0000	20C8	0008	0000 0000	20D8	0008	0000 0000	20E8	0008	0000 0000	20F8	0008	*0080 9243*
Data (8 bytes) MODULE_IDEN _TH_PARAM2			Data (8 bytes) MODULE_IDEN _TH_PARAM3			Data (8 bytes) MODULE_IDEN _TH_PARAM4			Data (6 bytes) MODULE_IDEN _COOL_TIME					
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value			
2108	0008	0080 0945	2118	0008	666E 3645	2128	0008	0000 0000	2185	0006	0078			

1) *MODULE_IDEN_TYPE* - variabe range 0-3, 0=NONE, 1=NOMEM, 2=1WIRE, 3=SIMPDC, Type: 2
2) *MODULE_IDEN_FIRM_VER* – version of firmware, FirmVer: 1
3) *MODULE_IDEN_HARD_VER* – version of hardware, HardVer: 1
4) *MODULE_IDEN_NAME* – module name, Name: “zasilacz 24v”

name	MODULE_IDEN_NAME																							
value	*0x20410024 7A6173696C61637A20323476000000000000000000000000000000000000000000*																							
byte no.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
hex	20	41	0	24	7A	61	73	69	6C	61	63	7A	20	32	34	76	0	0	0	0	0	0	0	0
symbol	0x2041		0x0024		z	a	s	i	l	a	c	z		2	4	v	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL
descrip.	OBJ_ID		size		ASCII characters																			

5) *MODULE_IDEN_SERIAL*- module serial number, 000000000-00
6) *MODULE_IDEN_DET_NAME* – detector name,

name	MODULE_IDEN_DET_NAME																							
value	*0x20610024 00000000000000000000000000000000000000000000000000000000000000000000*																							
byte no.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
hex	20	41	0	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
symbol	0x2061		0x0024		NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL	NUL
descrip.	OBJ_ID		size		ASCII characters																			

7) *MODULE_IDEN_DET_SERIAL* – detector serial number
8) *MODULE_IDEN_PROD_DATE* - date of manufacture of the module,|

```
struct {
        unsigned short int ms;
        unsigned char sec;
        unsigned char min;
        unsigned char hour;
        unsigned char day;
        unsigned char mon;
        unsigned char year;        //current year – 1900, rg. 2016 – 1900 = 116
}
```

milliseconds (2 bytes)	Second (1 byte)	Minutes (1 byte)	Hour (1 byte)	Day (1 byte)	Month (1 byte)	production year = value + 1900 (1 byte)
FFFF	FF	FF	FF	01	08	74
				1	VIII	2016 = 116 + 1900

0xFFFF|FF|FF|FF|01|08|74 = 01-08-2016

9) *MODULE_IDEN_TEC_TYPE* - variabe range 0-3, 0=NONE, 1=NOMEM, 2=1WIRE, 3=SIMPDC
10) *MODULE_IDEN_TH_TYPE* – type of thermistor
11) *MODULE_IDEN_TEC_PARAM1* – TEC parameter
12) *MODULE_IDEN_TEC_PARAM2* – TEC parameter
13) *MODULE_IDEN_TEC_PARAM3* – TEC parameter
14) *MODULE_IDEN_TEC_PARAM4* – TEC parameter
15) *MODULE_IDEN_TH_PARAM1* – thermistor parameter
16) *MODULE_IDEN_TH_PARAM2* – thermistor parameter
17) *MODULE_IDEN_TH_PARAM3* - thermistor parameter
18) *MODULE_IDEN_TH_PARAM4*- thermistor parameter
19) *MODULE_IDEN_COOL_TIME* – maximum time to cool a IR module. If the module does not reach desired temperature it will be turned off after 120s.

GET_MODULE_DEFAULT 2112 - command is used to read default configurations from module memory

Question command: *$084000047702#*
Response (MODULE_BASIC_PARAMS):

```
$24000033
2413000500
242400062EE0
24340006D120
2443000501
2453000500
246500060000
247400062EE0
2487000800038270
6255#
```

START $	ALL DATA			CRC-16bit	STOP #
	Head 'MODULE_BASIC _PARAMS'		DATA 1 - 8		
	OBJ_ID	size of all data			
24	2400	0033		6255	25

Data (5 bytes) MODULE_BASIC _PARAMS_SUP _CTRL			Data (6 bytes) MODULE_BASIC _PARAMS_U_SUP _PLUS			Data (6 bytes) MODULE_BASIC _PARAMS_U_SUP _MINUS			Data (5 bytes) MODULE_BASIC _PARAMS_FAN _CTRL		
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value
2413	0005	00	2424	0006	2EE0	2434	0006	D120	2443	0005	01
Data (5 bytes) MODULE_BASIC _PARAMS_TEC _CTRL			Data (6 bytes): MODULE_BASIC _PARAMS_PWM			Data (6 bytes): MODULE_BASIC _PARAMS_I_TEC _MAX			Data (8 bytes): MODULE_BASIC _PARAMS_T_DET		
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value
2453	0005	00	2465	0006	0000	2474	0006	2EE0	2487	0008	0003 8270

1) MODULE_BASIC_PARAMS_SUP_CTRL – variable is used to set operating mode of power supply output. Variable range 0...2. 0=AUTO, 1=OFF, 2=ON. AUTO mode is used to protect the detector. Power supply for preamplifier will be turn on when detector is cooled to desired temperature (GREEN LED indicator). OFF/ON means disable/enable power supply outputs.
2) MODULE_BASIC_PARAMS_U_SUP_PLUS – is used to set plus output voltage for pin 9 in DUBOX2x5 connector. Variable range 3000...15000 corresponds to 3...15V.
3) MODULE_BASIC_PARAMS_U_SUP_MINUS – is used to set minus output voltage for pin 6 in DUBOX2x5 connector. Variable range -15000...-3000 corresponds to -15...-3V.
*4) MODULE_BASIC_PARAMS_FAN_CTRL – f*an control output (Pin 7 DUBOX2x5 connector). Variable range 0...2. 0=AUTO, 1=OFF, 2=ON. Mode AUTO/ON and OFF mean fan enable and disable.
5) *MODULE_BASIC_PARAMS_TEC_CTRL* - Variable range 0...2. 0=AUTO, 1=OFF, 2=ON. Variable is used to set operating mode of TEC cooler.
6) *MODULE_BASIC_PARAMS_PWM* - PWM settings of TEC. Variable range 0...65535
7) *MODULE_BASIC_PARAMS_I_TEC_MAX* - maximum current for TEC. Variable range 0...20475 corresponds to 0...2.0475A.
8) MODULE_BASIC_PARAMS_T_DET - detector temperature in K degree. Variable range 0...400000 responds to 0...400 K, 18bits ADC.

The remaining three commands have the same data structure. For this reason they will not be described in detail. (*GET_MODULE_USER_SET 2144*, GET_MODULE_USER_MIN 2176, GET_MODULE_USER_MAX 2208)

GET_MODULE_USER_SET 2144 - command is used to read basic user settings from module memory

Question command: *$08600004BD03#*

Response (MODULE_BASIC_PARAMS):

$24000033
2413000500
242400062EE0
24340006D120
2443000501
2453000500
246500060000
247400062EE0
2487000800038270
6255#

GET_MODULE_USER_MIN 2176 – command is used to read minimum basic settings from module memory

Question command: *$088000044B02#*

Response (MODULE_BASIC_PARAMS):

$24000033
2413000500
242400062EE0
24340006D120
2443000501
2453000500
246500060000
247400060000
248700080002BF20
55BD#

GET_MODULE_USER_MAX 2208 - command is used to save minimum basic settings of no memory module

Question command: *$08A000048103#*

Response (MODULE_BASIC_PARAMS):

$24000033
2413000500
242400062EE0
24340006D120
2443000501
2453000500
246500060000
247400062EE0
24870008000493E0
9FE8#

“GET” COMMANDS DESCRIPTIONS FOR MODULE SMIPDC

In this chapter is a description of “GET” commands for SMIPDC module with build in memory. All configurations data is stored in SMIPDC module memory. Communications with 1 wire memory are done by pins: DATA 8, GND 3 in DUBOX2x5 connector (DS24B33).

GET_MODULE_SMIPDC_MONITOR 2560 - command is used to read actuall controller data configurations from module SMIPDC

Question command: *$0A0000041B02#*

Response (MODULE_SMIPDC_MONITOR):

$2C000046
2C1400060000
2C2400060000
2C3400060000
2C4400060000
2C5400060000
2C6400060000
2C7400060000
2C8400060000
2C9400060000
2CA400060000
2CB400060000
DCBC#

START $	ALL DATA			CRC-16bit	STOP #
	HEAD 'MODULE_SMIPDC _MONITOR'		DATA 1 - 8		
	OBJ_ID	size of all data			
24	2C00	0046		DCBC	25

Data (6 bytes): MODULE_SMIPDC_ MONITOR _SUP_PLUS			Data (6 bytes): MODULE_SMIPDC_ MONITOR_SUP _MINUS			Data (6 bytes): MODULE_SMIPDC_M ONITOR_FAN_PLUS			Data (6 bytes): MODULE_SMIPDC _MONITOR_TEC _PLUS		
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value
2C14	0006	0000	*2C24*	0006	0000	*2C34*	0006	0000	*2C44*	0006	0000
Data (6 bytes): MODULE_SMIPDC_ MONITOR_TEC_MI NUS			Data (6 bytes): MODULE_SMIPDC_ MONITOR_TH1			Data (6 bytes): MODULE_SMIPDC_M ONITOR_TH2			Data (6 bytes): MODULE_SMIPDC _MONITOR _U_DET		
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value
2C54	0006	0000	*2C64*	0006	0000	*2C74*	0006	0000	*2C84*	0006	0000
Data (6 bytes): MODULE_SMIPDC _MONITOR_U_1ST			Data (6 bytes): MODULE_SMIPDC _MONITOR _U_OUT			Data (6bytes): MODULE_SMIPDC _MONITOR_TEMP					
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value			
2C94	0006	0000	*2CA4*	0006	0000	*2CB4*	0006	0000			

1) MODULE_SMIPDC_MONITOR_SUP_PLUS -setting plus output voltage for pin 9 in DUBOX2x5 connector. Variable range 3000...15000 corresponds to 3...15V.
2) MODULE_SMIPDC_MONITOR_SUP_MINUS -setting minus output voltage for pin 6 in DUBOX2x5 connector. Variable range -15000...-3000 corresponds to -15...3V.
3) MODULE_SMIPDC_MONITOR_FAN_PLUS -Fan control output (Pin 7 DUBOX2x5 connector). Variable range 0...2. 0=AUTO, 1=OFF, 2=ON. Mode AUTO/ON/OFF mens fan enable. For safety cannot turn off the fan.
4) MODULE_SMIPDC_MONITOR_TEC_PLUS - maximum current for TEC. Variable range 0...20475 corresponds to 0...2.0475A.
5) MODULE_SMIPDC_MONITOR_TEC_MINUS - maximum current for TEC. Variable range 0...20475 corresponds to 0...2.0475A.
6) MODULE_SMIPDC_MONITOR_TH1 -
7) MODULE_SMIPDC_MONITOR_TH2 -
8) MODULE_SMIPDC_MONITOR_U_DET - -2048...2047 corresponds to -2.048...2.047V
9) MODULE_SMIPDC_MONITOR_U_1ST - -4096...4094 corresponds to -4.096...2.094V
10) MODULE_SMIPDC_MONITOR_U_OUT - -10.240...10.235 corresponds to -10.240...10.235V
11) MODULE_SMIPDC_MONITOR_TEMP - detector temperature in K degree. Variable range 0...400000 responds to 0...400 K, 18bits ADC.

GET_MODULE_SMIPDC_DEFAULT 2688 - command is used to read default configurations from module SMIPDC

Question command: *$0A800004F303#*
Response (MODULE_SMIPDC_PARAMS):

$30000031
301500060000
302500060000
303500060000
304500060000
305500060000
3063000500
3073000500
3083000500
DCE3#

START $	ALL DATA			CRC-16bit	STOP #
	HEAD 'MODULE_SMIPDC_PARAMS'		DATA 1 - 8		
	OBJ_ID	size of all data			
24	3000	0031		DCE3	25

Data (6 bytes): MODULE_SMIPDC_MONITOR_SUP_PLUS			Data (6 bytes): MODULE_SMIPDC_MONITOR_SUP_MINUS			Data (6 bytes): MODULE_SMIPDC_MONITOR_FAN_PLUS			Data (6 bytes): MODULE_SMIPDC_MONITOR_TEC_PLUS		
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value
3000	0006	0000	3025	0006	0000	3035	0006	0000	3045	0006	0000
Data (6 bytes) MODULE_SMIPDC_PARAMS_VARACTOR			Data (5 bytes): MODULE_SMIPDC_PARAMS_TRANS			Data (5 bytes): MODULE_SMIPDC_PARAMS_ACDC			Data (5 bytes): MODULE_SMIPDC_PARAMS_BW		
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value
3055	0006	00	3063	0005	00	3073	0005	00	3083	0005	00

1) MODULE_SMIPDC_PARAMS_DET_U - Variable range 0...256 corrseponds to 0-1V. Bias voltage
2) MODULE_SMIPDC_PARAMS_DET_I - Variable range 0...256 0-10mA. Bias current compensation.
3) MODULE_SMIPDC_PARAMS_GAIN - Variable range 0...256. Gain settings of the second stage.
4) MODULE_SMIPDC_PARAMS_OFFSET - Variable range 0...256 +1...-1V
5) MODULE_SMIPDC_PARAMS_VARACTOR - Variable range 0...4095
6) MODULE_SMIPDC_PARAMS_TRANS - Variable 0 or 1, 0=LOW 1=HIGH
7) MODULE_SMIPDC_PARAMS_ACDC - Variable 0 or 1, 0=AC 1=DC
8) MODULE_SMIPDC_PARAMS_BW - Variable range 0/1/2, 0=LOW, 1=MID, 2 HIGH

The remaining three commands have the same data structure. For this reason they will not be described in detail. (*GET_MODULE_SMIPDC_USER_SET 2720*, GET_MODULE_SMIPDC_USER_MIN 2752, GET_MODULE_SMIPDC_USER_MAX 2784)

GET_MODULE_SMIPDC_USER_SET 2720 - command is used to read configurations from module SMIPDC

Question command: $0AA000043902#
Response (MODULE_SMIPDC_PARAMS):

$30000031
301500060000
302500060000
303500060000
304500060000
305500060000
3063000500
3073000500
3083000500
DCE3#

GET_MODULE_SMIPDC_USER_MIN 2752 - command is used to read minimum settings from module SMIPDC

Question command: $0AC000042702#
Response (MODULE_SMIPDC_PARAMS):

$30000031
301500060000
302500060000
303500060000
304500060000
305500060000
3063000500
3073000500
3083000500
DCE3#

GET_MODULE_SMIPDC_USER_MAX 2784 - command is used to read maximum settings from module SMIPDC

Question command: $0AE00004ED03#
Response (MODULE_SMIPDC_PARAMS):

$30000031
301500060000
302500060000
303500060000
304500060000
305500060000
3063000500
3073000500
3083000500
DCE3#

“SET” COMMANDS DESCRIPTIONS

SET_SERVICE_MODE 1024 - command is used to set service mode.

Settings command:

$0410000D
10000009
101B000501
6F96#

START $	ALL DATA							CRC-16bit	STOP #
	HEAD 'SET_SERVICE_MODE'		VAR_DATA						
			HEAD 'SERVICE_MODE'		Data SERVICE_MODE_ENABLE				
	OBJ_ID	size of all data	OBJ_ID	size of var_data	OBJ_ID	size of data 1	value		
24	0410	000D	1000	0009	101B	0005	01	*6F96*	25

Response (SERVICE_MODE):

$10000009
101B000501
EEC8#

SERVICE_MODE_ENABLE - true/false value. Responsible for enabled/disabled service mode.

SET_TRANSPARENT_MODE 1104 - command is used to set transparent mode.

Settings command:

$0450000D
14000009
141B000501
5054#

START $	ALL DATA							CRC-16bit	STOP #
	HEAD 'SET_TRANSPARENT_MODE'		VAR_DATA						
			HEAD 'TRANSPARENT_MODE'		Data TRANSPARENT_MODE_ENABLE				
	OBJ_ID	size of all data	OBJ_ID	size of var_data	OBJ_ID	size of data 1	value		
24	0450	000D	1400	0009	*141B*	0005	01	*5054*	25

Response (SERVICE_MODE):

$14000009
141B000501
EE0B#

TRANSPARENT_MODE_ENABLE - true/false value. Responsible for enabled/disabled transparent mode.

SET_SMARTTEC_CONFIG 1296 - command is used to set configuration of SMARTEC type (Basic/OEM/Adavanced) and memory compatiblety.

Settings command:

$05100012
1800000E
1813000501
182B000500
DD84#

START $	CONTAINER 1				CRC-16bit	STOP #
	'SET_SMARTEC _CONFIG'		CONTAINER 2 'SMARTEC _CONFIG'			
	OBJ_ID	size of cont.1	OBJ_ID	size of cont. 2	DATA 1-2	
24	0510	0012	1800	000E	*DD84*	25

Data (5 bytes) SMARTTEC_ CONFIG_VARIANT			Data (6 bytes) SMARTTEC_ CONFIG_NO_MEM _COMPATIBLE		
OBJ_ID	size of data 1	value	OBJ_ID	size of data 2	value
1813	0005	01	182B	0005	00

Response (SMARTTEC_CONFIG):

$1800000E
1813000501
182B000500
D80B#

SMARTTEC_CONFIG_VARIANT - range 0...2. 0 = Basic, 1 = OEM , 2 = Advanced. Variable is used to determine the version of SMARTTEC controller.

SMARTTEC_CONFIG_NO_MEM_COMPATIBLE - true or false value.

SET_SMARTTEC_MOD_NO_MEM_IDEN 1552 - command is used to set data in no memory IR module (NOMEM)

Settings command:

$061000CD
200000C9
2013000500
202500060000
203500060001
2041002400000000000000000000000000000000000000000000000000000000000000000000
205A000800000000
2061002400000000000000000000000000000000000000000000000000000000000000000000
207A000800000000
2089000CFFFFFFFFFF010874
2093000501
20A3000501
20B8000800C0DA44
20C8000800007041

20D8000800000000
20E800800000000
20F8000800809243
2108000800800945
21180008666E3645
2128000800000000
218500060078
D5C6#

START $	CONTAINER 1					CRC-16bit	STOP #
	'SET_SMARTTEC_MOD_NO_MEM_IDEN'		CONTAINER 2				
			'MODULE_IDEN'		DATA 1-19		
	OBJ_ID	size of all data	OBJ_ID	size of var_data			
24	0610	00CD	2000	00C9		*D5C6*	25

Data (1 byte) MODULE_IDEN _TYPE			Data (2 bytes) MODULE_IDEN _FIRM_VER			Data (2 bytes) MODULE_IDEN _HARD_VER			Data (20 bytes) MODULE_IDEN _NAME			Data (4 bytes) MODULE_IDEN _SERIAL		
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value
2013	0005	00	2025	0006	0000	2035	0006	01	2041	0024	table below	205A	0008	0000 0000

Data (20 bytes) MODULE_IDEN _DET_NAME			Data (4 bytes) MODULE_IDEN _DET_SERIAL			Data (8 bytes) MODULE_IDEN _PROD_DATE			Data (1 bytes) MODULE_IDEN _TEC_TYPE			Data (bytes) MODULE_IDEN _TH_TYPE		
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value
2061	0024	table below	207A	0008	0000 0000	2089	000C	table below	2093	0005	01	20A3	0005	01

Data (4 bytes) MODULE_IDEN _TEC_PARAM1			Data (4 bytes) MODULE_IDEN _TEC_PARAM2			Data (4 bytes) MODULE_IDEN _TEC_PARAM3			Data (4 bytes) MODULE_IDEN _TEC_PARAM4			Data (4 bytes) MODULE_IDEN _TH_PARAM1		
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value
20B8	0008	00C0D A44	20C8	0008	0000 7041	20D8	0008	0000 0000	20E8	0008	0000 0000	20F8	0008	0080 9243

Data (4 bytes) MODULE_IDEN _TH_PARAM2			Data (4 bytes) MODULE_IDEN _TH_PARAM3			Data (4 bytes) MODULE_IDEN _TH_PARAM4			Data (2 bytes) MODULE_IDEN| _COOL_TIME		
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value
2108	0008	0080 0945	2118	0008	666E 3645	2128	0008	0000 0000	2185	0006	0078

Response (MODULE_IDEN):

$200000C9
2013000500
202500060000
203500060001
204100240000000000000000000000000000000000000000000000000000000000000000
205A000800000000
206100240000000000000000000000000000000000000000000000000000000000000000
207A000800000000

2089000CFFFFFFFFFF010874
209300050120A3000501
20B8000800C0DA44
20C8000800007041
20D8000800000000
20E8000800000000
20F8000800809243
2108000800800945
21180008666E3645
2128000800000000
218500060078
773B#

SET_SMARTTEC_MOD_NO_MEM_DEFAULT 1584 - command is used to set default configuration in no memory IR module (NOMEM)

Settings command:

$06300037
24000033
2413000500
242400062328
24340006DCD8
2443000500
2453000500
246500060000
247400061194
2487000800038270
77B0#

START $	CONTAINER 1					CRC-16bit	STOP #
	SET_SMARTTEC_MOD_NO_MEM_IDEN		CONTAINER 2				
			MODULE_IDEN				
	OBJ_ID	size of all data	OBJ_ID	size of var_data	DATA 1-8		
24	0630	0037	2400	0033		*77B0*	25

Data (1 byte) MODULE_IDEN _TYPE			Data (2 bytes) MODULE_IDEN _FIRM_VER			Data (2 bytes) MODULE_IDEN _HARD_VER			Data (1 byte) MODULE_IDEN _NAME		
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value
2413	0005	00	2424	0006	2328	2434	0006	DCD8	2443	0005	00
Data (1 byte) MODULE_IDEN _TYPE			Data (2 bytes) MODULE_IDEN _FIRM_VER			Data (2 bytes) MODULE_IDEN _HARD_VER			Data (2 bytes) MODULE_IDEN _NAME		
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value
2453	0005	00	2465	0006	0000	2474	0006	1194	2487	0008	0003 8270

Response (MODULE_BASIC_PARAMS):

$24000033
2413000500
242400062328
24340006DCD8

2443000500
2453000500
246500060000
247400061194
2487000800038270
7562#

The remaining three commands have the same data structure. For this reason they will not be described in detail. (SET_SMARTTEC_MOD_NO_MEM_USER_SET 1616, SET_SMARTTEC_MOD_NO_MEM_USER_MIN 1648, SET_SMARTTEC_MOD_NO_MEM_USER_MAX 1680)

SET_SMARTTEC_MOD_NO_MEM_USER_SET 1616 - command is used to set user configuration in no memory IR module (NOMEM)

Settings command:

$06500037
24000033
2413000500
242400062328
24340006DCD8
2443000500
2453000500
246500060000
247400061194
2487000800038270
4A65#

Response (MODULE_BASIC_PARAMS):

$24000033
2413000500
242400062328
24340006DCD8
2443000500
2453000500
246500060000
247400061194
2487000800038270
7562#

SET_SMARTTEC_MOD_NO_MEM_USER_MIN 1648 - command is used to set user lower limits configuration in no memory IR module (NOMEM)

Settings command:

$06700037
24000033
2413000500
242400060BB8
24340006C568
2443000500
2453000500
246500060000
247400060000
248700080002BF20
0AEA#

Response (MODULE_BASIC_PARAMS):

$24000033
2413000500

242400060BB8
24340006C568
2443000500
2453000500
246500060000
247400060000
248700080002BF20
215E#

SET_SMARTTEC_MOD_NO_MEM_USER_MAX 1680 - command is used to set user upper limits configuration in no memory IR module (NOMEM)

Settings command:

$06900037
24000033
2413000500
242400063A98
24340006F448
2443000500
2453000500
246500060000
247400062EE0
24870008000493E0
3096#

Response (MODULE_BASIC_PARAMS):

$24000033
2413000500
242400063A98
24340006F448
2443000500
2453000500
246500060000
247400062EE0
24870008000493E0
743B#

SET_MODULE_IDEN 2064 - command is used to set data configuration
Settings command:

$081000CD
200000C9
20130005FF
20250006FFFF
20350006FFFF
20410024FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF00
205A0008FFFFFFFF
20610024FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF00
207A0008FFFFFFFF
2089000CFFFFFFFFFFFFFFFF
20930005FF
20A30005FF
20B80008FFFFFFFF
20C80008FFFFFFFF
20D80008FFFFFFFF
20E80008FFFFFFFF
20F80008FFFFFFFF
21080008FFFFFFFF
21180008FFFFFFFF
21280008FFFFFFFF

21850006FFFF
3AA6#

<table>
<tr><td rowspan="4">START
$</td><td colspan="5">CONTAINER 1</td><td rowspan="4">CRC-16bit</td><td rowspan="4">STOP
#</td></tr>
<tr><td colspan="2" rowspan="2">'SET_SMARTTEC_MOD_NO_MEM_IDEN'</td><td colspan="3">CONTAINER 2</td></tr>
<tr><td colspan="2">'MODULE_IDEN'</td><td rowspan="3">DATA 1-19</td></tr>
<tr><td>OBJ_ID</td><td>size of all data</td><td>OBJ_ID</td><td>size of var_data</td></tr>
<tr><td>24</td><td>0610</td><td>00CD</td><td>2000</td><td>00C9</td><td>D5C6</td><td>25</td></tr>
</table>

MODULE_IDEN _TYPE			MODULE_IDEN _FIRM_VER			MODULE_IDEN _HARD_VER			MODULE_IDEN _NAME			MODULE_IDEN _SERIAL		
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value
2013	0005	00	2025	0006	0000	2035	0006	01	2041	0024	table below	205A	0008	0000 0000
MODULE_IDEN _DET_NAME			MODULE_IDEN _DET_SERIAL			MODULE_IDEN _PROD_DATE			MODULE_IDEN _TEC_TYPE			MODULE_IDEN _TH_TYPE		
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value
2061	0024	table below	207A	0008	0000 0000	2089	000C	table below	2093	0005	01	20A3	0005	01
MODULE_IDEN _TEC_PARAM1			MODULE_IDEN _TEC_PARAM2			MODULE_IDEN _TEC_PARAM3			MODULE_IDEN _TEC_PARAM4			MODULE_IDEN _TH_PARAM1		
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value
20B8	0008	00C0 DA44	20C8	0008	0000 7041	20D8	0008	0000 0000	20E8	0008	0000 0000	20F8	0008	0080 9243
MODULE_IDEN _TH_PARAM2			MODULE_IDEN _TH_PARAM3			MODULE_IDEN _TH_PARAM4			MODULE_IDEN _COOL_TIME					
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value			
2108	0008	0080 0945	2118	0008	666E 3645	2128	0008	0000 0000	2185	0006	0078			

Response (MODULE_IDEN):

$200000C9
20130005FF
20250006FFFF
20350006FFFF
20410024FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
205A0008FFFFFFFF
20610024FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
207A0008FFFFFFFF
2089000CFFFFFFFFFFFFFFFF
20930005FF20A30005FF
20B80008FFFFFFFF
20C80008FFFFFFFF
20D80008FFFFFFFF
20E80008FFFFFFFF
20F80008FFFFFFFF
21080008FFFFFFFF

21180008FFFFFFFF
21280008FFFFFFFF
21850006FFFF
84AE#

SET_MODULE_DEFAULT 2128 - command is used to set default data configuration
Settings command:

$08500037
24000033
24130005FF
24240006FFFF
24340006FFFF
24430005FF
24530005FF
24650006FFFF
24740006FFFF
24870008FFFFFFFF
01FD#

START $	CONTAINER 1					CRC-16bit	STOP #
	HEAD 'SET_SMARTTEC_MOD_NO_MEM_IDEN'		CONTAINER 2				
	OBJ_ID	size of all data	HEAD 'MODULE_IDEN'		DATA		
			OBJ_ID	size of var_data			
24	0850	0037	2400	0033		01FD	25

MODULE_IDEN_TYPE			MODULE_IDEN_FIRM_VER			MODULE_IDEN_HARD_VER			MODULE_IDEN_NAME		
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value
2413	0005	00	2424	0006	2328	2434	0006	DCD8	2443	0005	00
MODULE_IDEN_TYPE			MODULE_IDEN_FIRM_VER			MODULE_IDEN_HARD_VER			MODULE_IDEN_NAME		
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value
2453	0005	00	2465	0006	0000	2474	0006	1194	2487	0008	0003 8270

Response (MODULE_BASIC_PARAMS):

$24000033
24130005FF
24240006FFFF
24340006FFFF
24430005FF
24530005FF
24650006FFFF
24740006FFFF
24870008FFFFFFFF
BA51#

The remaining three commands have the same data structure. For this reason they will not be described in detail. (SET_MODULE_USER_SET 2160, SET_MODULE_USER_MIN 2192, SET_MODULE_USER_MAX 2224)

SET_MODULE_USER_SET 2160 - command is used to set user configuration

Settings command:

```
$08700037
24000033
24130005FF
24240006FFFF
24340006FFFF
24430005FF
24530005FF
24650006FFFF
24740006FFFF
24870008FFFFFFFF
154E#
```

Response (MODULE_BASIC_PARAMS):

```
$24000033
24130005FF
24240006FFFF
24340006FFFF
24430005FF
24530005FF
24650006FFFF
24740006FFFF
24870008FFFFFFFF
BA51#
```

SET_MODULE_USER_MIN 2192 - command is used to set user lower limits configuration

Settings command:

```
$08900037
24000033
24130005FF
24240006FFFF
24340006FFFF
24430005FF
24530005FF
24650006FFFF
24740006FFFF
24870008FFFFFFFF
7A57#
```

Response (MODULE_BASIC_PARAMS):

```
$24000033
24130005FF
24240006FFFF
24340006FFFF
24430005FF
24530005FF
24650006FFFF
24740006FFFF
24870008FFFFFFFF
BA51#
```

SET_MODULE_USER_MAX 2224 - command is used to set user lower limits configuration

Settings command:

$08B00037
24000033
24130005FF
24240006FFFF
24340006FFFF
24430005FF
24530005FF
24650006FFFF
24740006FFFF
24870008FFFFFFFF
6EE4#

Response (MODULE_BASIC_PARAMS):

$24000033
24130005FF
24240006FFFF
24340006FFFF
24430005FF
24530005FF
24650006FFFF
24740006FFFF
24870008FFFFFFFF
BA51#

SET_MODULE_SMIPDC_DEFAULT 2704 - command is used to set default configuration in SMIPDC module

Settings command:

$0A900035
30000031
301500060000
302500060000
303500060000
304500060000
305500060000
3063000500
3073000500
3083000500
6C00#

START $	CONTAINER 1					CRC-16bit	STOP #
	'SET_SMARTTEC_MOD_NO_MEM_IDEN'		CONTAINER 2				
	OBJ_ID	size of all data	'MODULE_IDEN' OBJ_ID	size of var_data	DATA		
24	0A90	0035	3000	0031		6C00	25

MODULE_SMIPDC_PARAMS_DET_U			MODULE_SMIPDC_PARAMS_DET_I			MODULE_SMIPDC_PARAMS_GAIN			MODULE_SMIPDC_PARAMS_OFFSET		
OBJ_ID	size of	value	OBJ_ID	size of	value	OBJ_ID	size of	value	OBJ_ID	size of	value

	data			data			data			data	
2413	0005	00	2424	0006	2328	2434	0006	DCD8	2443	0005	00
MODULE_SMIPDC _PARAMS _VARACTOR			MODULE_SMIPDC _PARAMS_TRANS			MODULE_SMIPDC _PARAMS_ACDC			MODULE_SMIPDC _PARAMS_BW		
OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value	OBJ_ID	size of data	value
2453	0005	00	2465	0006	0000	2474	0006	1194	2487	0008	0003 8270

Response (MODULE_SMIPDC_PARAMS):

$30000031
301500060000
302500060000
303500060000
304500060000
305500060000
3063000500
3073000500
3083000500
DCE3#

The remaining two commands have the same data structure. For this reason they will not be described in detail. (SET_MODULE_SMIPDC_USER_SET 2736, SET_MODULE_SMIPDC_USER_MIN 2768, SET_MODULE_SMIPDC_USER_MAX 2800)

SET_MODULE_SMIPDC_USER_SET 2736 - command is used to set user configuration in SMIPDC module

Settings command:

$0AB00035
30000031
301500060000
302500060000
303500060000
304500060000
305500060000
3063000500
3073000500
3083000500
DDBA#

Response (MODULE_SMIPDC_PARAMS):

$30000031
301500060000
302500060000
303500060000
304500060000
305500060000
3063000500
3073000500
3083000500
DCE3#

SET_MODULE_SMIPDC_USER_MIN 2768 - command is used to set user lower limits configuration in SMIPDC module

Settings command:

$0AD00035
30000031
301500060000
302500060000
303500060000
304500060000
305500060000
3063000500
3073000500
3083000500
4F77#

Response (MODULE_SMIPDC_PARAMS):

$30000031
301500060000
302500060000
303500060000
304500060000
305500060000
3063000500
3073000500
3083000500
DCE3#

SET_MODULE_SMIPDC_USER_MAX 2800 - command is used to set user upper limits configuration in SMIPDC module

Settings command:

$0AF00035
30000031
301500060000
302500060000
303500060000
304500060000
305500060000
3063000500
3073000500
3083000500
FECD#

Response (MODULE_SMIPDC_PARAMS):

$30000031
301500060000
302500060000
303500060000
304500060000
305500060000
3063000500
3073000500
3083000500
DCE3#